The Weak Base Method for Constraint Satisfaction

Von der

Fakultät für Elektrotechnik und Informatik
der Gottfried Wilhelm Leibniz Universität Hannover

zur Erlangung des Grades einer

DOKTORIN DER NATURWISSENSCHAFTEN

Dr. rer. nat.

genehmigte Dissertation
von

Dipl.-Math. Ilka Schnoor

geboren am 11. September 1979, in Eckernförde

2008

Schlagwörter: Berechnungskomplexität, Constraint Satisfaction Probleme, Universelle Algebra

Keywords: Computational complexity, Constraint satisfaction problems, Universal algebra

Referent: Heribert Vollmer, Gottfried Wilhelm Leibniz Universität Hannover
Korreferent: Martin Mundhenk, Friedrich-Schiller-Universitt Jena
Tag der Promotion: 21.11.2007

Danksagung

Vor kurzem, so scheint es, begann ich erst mein Studium und nun habe ich diese Arbeit fertiggestellt. In dieser Zeit haben mich viele Menschen auf diesem Weg begleitet und unterstützt. Zuerst sind da meine Betreuer zu nennen: Heribert Vollmer, der mich angeleitet und ermutigt hat und mich auf einer Stelle getragen hat, die es eigentlich gar nicht gab, und Nadia Creignou, die unermüdlich Verbesserungsvorschläge für diese Arbeit geliefert und viele Formulare für mich ausgefüllt hat. Dann meine Kollegen Michael Bauland und Anca Vais, die durch ihre lebhafte Art immer wieder Schwung ins Institut bringt und mit der ich gerne ein Büro geteilt habe. Und alle meine Coautoren.

Ganz besonders Erwähnen möchte ich meine Eltern Frauke und Jürgen Johannsen, die mir die wissenschaftliche Ausbildung ermöglicht und mich dabei immer unterstützt haben. Und natürlich Henning, der so vieles für mich ist: Freund, Kollege und mein Mann, und der alle Höhen und Tiefen in der Entstehung dieser Arbeit mit mir durchlebt hat.

Euch allen vielen Dank!

Acknowledgement

Not long ago, it seems, I started to study Mathematics and now I have completed this thesis. Many were with me in this time and gave me support. My advisors are the first ones to mention: Heribert Vollmer, who directed and encouraged me and managed to employ me without a position, and Nadia Creignou, who gave many constructive suggestions for this thesis and did a lot of paperwork. My colleagues Michael Bauland and Anca Vais, who brought live to the institute and with whom I liked to share an office. And all my coauthors.

Especially I want to mention my parents Frauke and Jürgen Johannsen, who made my scientific education possible and always fully supported me. And of course Henning, who is so much to me: friend, collegue, and husband, and who went with me through all highs and lows I experienced during the writing of this thesis.

I thank you all very much!

Zusammenfassung

Constraint Satisfaction Probleme sind von großer Bedeutung in der Komplexitätstheorie. Sie verallgemeinern eine Vielzahl von Erfüllbarkeits- und kombinatorischen Problemen und liefern kanonische Repräsentanten für viele Komplexitätsklassen. Schaefer klassifizierte in [Sch78] die Komplexität Boolscher Constraint Satisfaction Probleme und zeigte ein dichotomes Komplexitätsverhalten dieser Probleme, welches auch für Constraint Satisfaction Probleme über beliebigen endlichen Grundbereichen vermutet wird [FV98]. Algebraische Werkzeuge, die eine Galois-Verbindung zwischen in Constraint-Instanzen auftretenden Klauseln und Mengen von Funktionen ausnutzen, liefern eine Methode um Komplexitätsklassifikationen für Constraint-Probleme zu finden und zu beweisen. Es gibt jedoch viele zu Constraint Satisfaction verwandte Probleme, für die diese Methode nicht benutzt werden kann.

In dieser Arbeit entwickeln wir eine Methode die es ermöglicht eine verfeinerte Galois-Verbindung zu benutzen um Komplexitätsklassifikationen für solche Probleme zu erhalten. Anschließend führen wir diese Methode vor, indem wir die Komplexität zweier aus verschiedenen Bereichen stammender Constraint-Probleme klassifizieren. Zuerst betrachten wir das balancierte Erfüllbarkeits-Problem, bei dem nach Lösungen gefragt wird, die, zusätzlich zu den in der Constraint-Instanz gegebenen lokalen Bedingungen, eine globale Ausgewogenheits-Bedingung erfüllen. Dann beschäftigen wir uns mit einer nicht-monotonen Logik und untersuchen Fragestellungen für auf Constraint-Formeln beschränkte Default Logik. In beiden Fällen erzielen wir durch den Einsatz unserer neuen Methode vollständige Komplexitätsklassifikationen.

Abschließend untersuchen wir das Problem alle Lösungen einer gegebenen Constraint-Instanz aufzuzählen. Für Boolesche Instanzen wurde die Komplexität dieses Problems vollständig von Creignou und Hébrard klassifiziert [CH97]. Wir betrachten Instanzen über beliebigen endlichen Grundbereichen und präsentieren eine Familie von neuen effizienten Aufzähl-Algorithmen. Wir unternehmen außerdem erste Schritte auf dem Weg zu einer vollständigen Klassifikation für das Aufzähl-Problem über dem 3-wertigen Grundbereich.

Abstract

Constraint satisfaction problems are an important class of problems in complexity theory. They generalize many combinatorial problems as well as satisfiability problems and provide canonical complete problems for many complexity classes. The computational complexity of all Boolean constraint satisfaction problems was classified by Schaefer [Sch78] and reveals a dichotomic behavior that is conjectured to also hold for arbitrary domains [FV98]. Algebraic tools involving a Galois correspondence between clauses appearing in the constraint instances and sets of functions give a method to obtain complexity classifications in the constraint context. However, for many problems related to constraint satisfaction these tools cannot be applied.

In this thesis we develop a method that allows to use a refined Galois correspondence to obtain complexity classifications for those problems. Afterwards we demonstrate our new method by classifying two constraint problems from different contexts: first we consider the balanced satisfiability problem, where we require the solutions to satisfy a global condition additionally to the local constraints given in the constraint instance. Then we turn to nonmonotonic logics and study the complexity of reasoning in default logic restricted to constraint formulas. In both cases we achieve full classifications using our new method as an essential tool.

Finally we study the problem of enumerating all solutions of a given constraint instance. For the Boolean case a full classification has been achieved by Creignou and Hébrard [CH97]. We look at instances over arbitrary finite domains and present a template for new efficient enumeration algorithms. We achieve a first step towards a classification of the enumeration problem over the three-element domain.

Contents

List of Figures

Chapter 1

Introduction

In complexity theory we deal with the question what amount of resources computational tasks need to be completed by a computer. The resources we are interested in are time, i.e., the number of computation steps, and space, i.e., the required memory to perform the task. To argue on a formal basis in this field and independent from engineering details and programming languages, mathematical models of computers and algorithms are used. The most employed model is the Turing Machine introduced by Alan M. Turing in 1936 [Tur36]. Turing Machines allow a formal definition of *efficient algorithms* which need polynomial time in the length of the input to finish their computation. The class of all computational questions that can be answered by efficient algorithms is the class P.

In complexity theory, both positive and negative complexity results are of interest. We take a look at the *satisfiability problem*, which is the following question: we are given a propositional formula and have to find out whether this formula is satisfiable. Up to now no efficient algorithm for this task is known, nor is there a proof that such an algorithm does not exist.

But if we are additionally given an assignment for the formula, then we can easily verify whether it satisfies the formula or not. That means, even if we maybe cannot efficiently determine if the formula has a solution, we can efficiently verify the correctness of a given solution. This behavior is characteristic for the problems in the well-known class NP. As mentioned before, the satisfiability problem is not proven to be not in P, and so is no problem from NP. The question whether there indeed is an NP-problem that is not in P, or whether P equals NP, is the famous P-NP-problem and one of the most important open questions in complexity theory.

The hardest problems in NP are called NP-complete. They have the property, that an efficient algorithm solving such a problem directly yields efficient algorithms for all other NP-problems and thus implies that P equals NP. The

theory of NP-completeness was initiated by Stephen A. Cook in 1971, when he showed that the satisfiability problem, as first problem at all, is NP-complete [Coo71]. From then on many other problems have been proven to be NP-complete. A large collection of NP-complete problems was compiled by Michael R. Garey and David S. Johnson in [GJ79].

As the satisfiability problem naturally represents NP, many complexity classes are represented by problems related to formulas. The study of these *formula problems* helps to provide a better understanding of the structure of and relationships between complexity classes.

An important topic in this line is the study of *constraint satisfaction problems*. These are satisfiability problems for formulas that are of the form

$$C_1 \wedge \cdots \wedge C_n,$$

where $C_1, \ldots, C_n$ are clauses built from templates out of a fixed set Γ. The templates formalize constraints that are applied to variables in the clauses, so a satisfying assignment must take all those constraints of the variables formulated in the clauses into account.

Thomas J. Schaefer studied the complexity of the Boolean satisfiability problem and found out in 1978 that for every fixed and finite set of templates Γ, the problem is in P or NP-complete [Sch78]. This dichotomic behavior is surprising, because, due to a classic result from Richard E. Ladner, there are infinitely many degrees of complexity petween P and NP, provided that P differs from NP [Lad75]. Thomás Feder and Moshe Y. Vardi conjectured in [FV98], that this dichotomy also holds for non-Boolean constraint satisfaction problems, where a finite set of values can be assigned to the variables. This conjecture is still an open question and an active field of research. However, Andrei A. Bulatov succeeded in 2002 to prove that the dichotomy holds in the case of three-valued satisfaction problems [Bul06].

Besides satisfiability, many other problems have been studied, especially in the context of Boolean constraint problems. Nadia Creignou and Miki Hermann achieved a dichotomy for the problem of counting [CH96] and Nadia Creignou and Jean-J. Hébrard for the problem of enumerating [CH97] all solutions of a given Boolean constraint formula and there are classifications for the equivalence and the isomorphism problem by Elmar Böhler, Edith Hemaspaandra, Steffen Reith and Heribert Vollmer [BHRV02, BHRV04].

Constraint formulas also have been used to examine restrictions of nonmonotonic logics, which are widely used in artificial intelligence. In contrast to classical propositional logic, in nonmonotonic logics the set of consequences drawn from a set of formulas can shrink when adding more formulas. Some nonmonotonic logics investigated in the constraint context are circumscription,

studied by Gustav Nordh and Peter Johnsson [NJ04, Nor05], and abduction, studied by Nadia Creignou, Gustav Nordh, and Bruno Zanuttini [NZ05, CZ06].

One of the most effective techniques used to obtain complexity classifications for constraint problems are algebraic tools that group the infinitely many problems that arise from infinitely many template-sets Γ into a (still infinite but) well structured hierarchy. These tools were known in mathematics since the 1960's, and they were first applied in a complexity setting by Peter G. Jeavons, David A. Cohen, and Marc Gyssens in [JCG97, Jea98]. This technique allows an easy proof for Schaefer's dichotomy and was essential for Bulatov's dichotomy. However, it could not be used for all results mentioned, for example the complexity classifications for the enumerating problem by Creignou and Hébrard and for the equivalence problem by Böhler et al. needed alternative arguments. The reason is, that the classes in the hierarchy mentioned above are well suited for the satisfiability problem but do not canonically fit to the equivalence and the enumeration problem.

This is the starting point for the research in this thesis: we present an algebraic tool that allows to use a refined hierarchy suitable to most of all problems dealing with constraint formulas: we introduce *weak bases* giving a method which does not require to consider the refined classes in the hierarchy explicitly. Using this method it is often sufficient to investigate the well-known classes arising from the classical tools.

After presenting our method in Chapter 3, we show in the next two chapters how to use it. In Chapter 4 we study the complexity of the balanced constraint satisfiability problem, which is the question whether a given constraint formula has a satisfying assignment that is equally equipped with 0 and 1. This problem is related to the question whether a given formula has a satisfying assignment where the number of 1s is exactly a given number k. This question is of practical importance if every variable set to 1 is connected with a certain cost. We additionally consider the counting variants of both questions, and achieve a full complexity classification with respect to the fixed template-set Γ, for all of these problems. The key idea in proving the classification is the application of the weak base method developed in Chapter 3.

In Chapter 5 we apply our new method to computational questions arising in Default logic, a nonmonotonic logic introduced by Raymond Reiter. By making use of weak bases we accomplish complexity classifications that seemed not to be possible before.

Finally in Chapter 6 we consider the task to enumerate all solutions of a given constraint formula. Creignou and Hébrard gave a simple criterion determining exactly when there is an efficient enumeration algorithm in the Boolean case [CH97]. We show that this criterion cannot be generalized to non-Boolean enumeration problems. We present a broad class of new enumerating

algorithms and achieve a first step towards a full classification in the three-valued case.

Publications

The results in Chapter 3 will be published in [SS07b], an early version appeared as [SS06b]. Chapter 4 is based on unpublished joint work with Nadia Creignou and Henning Schnoor. The results in Chapter 5 are a continuation of the resarch in [CHS07], where we achieved a classification for what we there called "conjunctive queries." In this thesis we use the weak base method to obtain a classification for actual constraint formulas. Finally the results in Chapter 6 appeared in [SS07a].

Chapter 2

Preliminaries

2.1 Computational Complexity

We start with a few notations: $\mathbb{N}$ denotes the set of all natural numbers $\{0, 1, 2, \dots\}$. For a set D we define $D^1 = D$ and $D^{i+1} = D^i \times D$, $i \geq 1$. If $t \in D^k$ is some tuple, then $t[i]$ denotes the i-th component of t. If A is a subset of D we write $A \subseteq D$ and if the subset is strict we write $A \subsetneq D$.

We now introduce basic concepts from computational complexity. Let Σ be a finite alphabet and A a language over Σ, i.e., $A \subseteq \Sigma^* = \bigcup_{n \in \mathbb{N}} \Sigma^n$. Then A is a *decision problem* connected with the following question: Given a word $w \in \Sigma^*$, does w belong to A? We denote the complement of A by $\overline{A} =_{def} \Sigma^* \setminus A$.

As model of computation we use the Turing Machine (see [Pap94]). We will not introduce Turing Machines formally, since for the complexity classes we work with, we do not need a detailed view on their mode of computation.

We define complexity classes for decision problems: the class P is the set of all decision problems that can be decided by a deterministic Turing Machine in time bounded by a polynomial in the size of the input, NP is the set of all decision problems that can be decided by a non-deterministic Turing Machine in time bounded by a polynomial in the size of the input, and PSPACE is the class of all decision problems that can be decided by a deterministic Turing Machine in space bounded by a polynomial in the size of the input.

For a complexity class $\mathcal{C}$ we define the class co$\mathcal{C}$ to be the set $\{A \mid \overline{A} \in \mathcal{C}\}$. If $\mathcal{C}$ is defined by deterministic Turing Machines bounded in time or space, then co$\mathcal{C} = \mathcal{C}$, in particular it holds that coP = P and coPSPACE = PSPACE. However, it is unknown whether NP equals coNP.

Directly from the above definitions it follows that P $\subseteq$ NP and P $\subseteq$ PSPACE. The equation NP $\subseteq$ PSPACE can easily be seen as well.

Between P and PSPACE there lies the polynomial hierarchy, introduced by

Meyer and Stockmeyer [MS72, Sto77]. Its classes are defined via Turing machines with access to *oracles*. An oracle for a problem A can answer questions of the form "is w from A?" in constant time. For a class of problems $\mathcal{C}$ the classes $\mathrm{P}^{\mathcal{C}}$ and $\mathrm{NP}^{\mathcal{C}}$ consist of all decision problems solvable by a deterministic (and non-deterministic respectively) Turing machine with access to an oracle from $\mathcal{C}$ in polynomial time. Intuitively a problem from $\mathrm{P}^{\mathcal{C}}$ can be decided in polynomial time, if we have full knowledge about the problems in $\mathcal{C}$.

The polynomial hierarchy contains the following classes:

$$\Sigma_0^{\mathrm{P}} =_{def} \Delta_0^{\mathrm{P}} =_{def} \mathrm{P}$$

and for every $i \in \mathbb{N}$

$$\begin{aligned}
\Delta_{i+1}^{\mathrm{P}} &=_{def} \mathrm{P}^{\Sigma_i^{\mathrm{P}}},\\
\Sigma_{i+1}^{\mathrm{P}} &=_{def} \mathrm{NP}^{\Sigma_i^{\mathrm{P}}},\\
\Pi_i^{\mathrm{P}} &=_{def} \mathrm{co}\Sigma_i^{\mathrm{P}},\\
\mathrm{PH} &=_{def} \bigcup_{i\in\mathbb{N}} \Sigma_i^{\mathrm{P}}.
\end{aligned}$$

It is easy to see that $\Sigma_1^{\mathrm{P}} = \mathrm{NP}$ and $\Pi_1^{\mathrm{P}} = \mathrm{coNP}$. Furthermore the following inclusions hold for every $i \in \mathbb{N}$:

$$\Delta_i^{\mathrm{P}} \subseteq \Sigma_i^{\mathrm{P}} \subseteq \Delta_{i+1}^{\mathrm{P}} \subseteq \Pi_{i+1}^{\mathrm{P}} \subseteq \Delta_{i+2}^{\mathrm{P}} \subseteq \mathrm{PH} \subseteq \mathrm{PSPACE}.$$

This gives an inclusion structure as it is illustrated in Figure 2.1. For none of these inclusions it is known, whether it is strict, a special case is the famous question whether P equals NP. However, all inclusions are believed to be strict and the equality of Δ_k^{P} and Σ_k^{P} for some $k \in \mathbb{N}$ would imply $\Delta_k^{\mathrm{P}} = \Sigma_i^{\mathrm{P}} = \Pi_i^{\mathrm{P}} = \Delta_i^{\mathrm{P}} = \mathrm{PH}$ for all $i \geq k$.

The problems we study in this thesis are located in the first two levels of the polynomial hierarchy: the classes arising in our classifications are P, NP, coNP, Π_2^{P}, and Σ_2^{P}.

To compare the complexity of problems, many different notions of reductions have been introduced. For our purposes the following is suitable.

Let $A \subseteq \Sigma^*$ and $B \subseteq \Pi^*$ be decision problems, then A is *many-one reducible to B in logarithmic space* ($A \leq_m^{\log} B$) if there is a function $f : \Sigma^* \to \Pi^*$ that satisfies the following conditions:

1. for every $x \in \Sigma^*$ holds: $x \in A$ if and only if $f(x) \in B$, and
2. f can be computed by a deterministic Turing machine in space bounded by $c \log n + d$, where n is the length of the input and $d, c \in \mathbb{N}$ are constants.

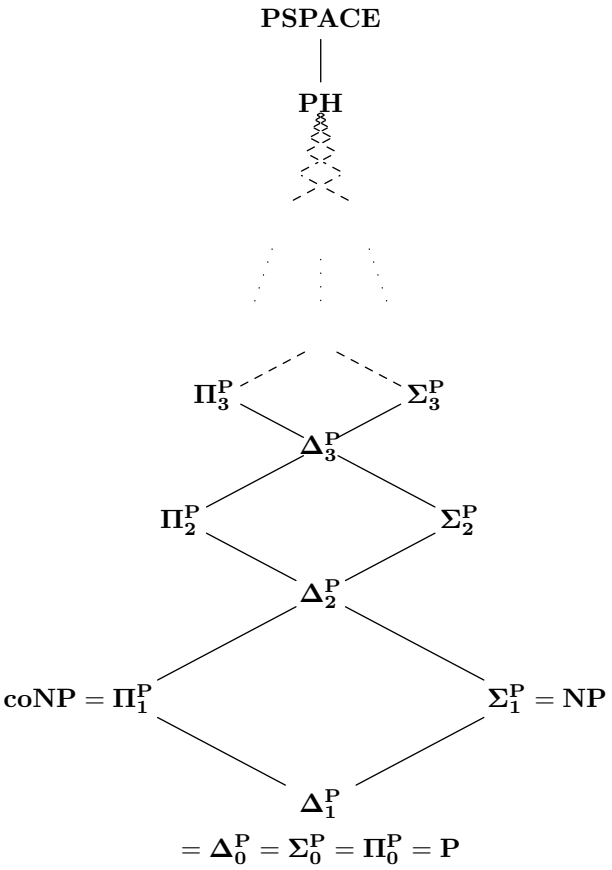

Figure 2.1: The Polynomial Time Hierarchy

Intuitively $A \leq_m^{\log} B$ means "A is not harder than B", because to decide whether $x \in A$ we can use a decision algorithm for B and ask whether $f(x) \in B$.

Let $\mathcal{C}$ be some complexity class. A decision problem A is called *hard for $\mathcal{C}$ (or $\mathcal{C}$-hard) under $\leq_m^{log}$*, if for every $B \in \mathcal{C}$ it holds $B \leq_m^{\log} A$, i.e., if no problem from $\mathcal{C}$ is harder than A. If additionally it holds that $A \in \mathcal{C}$, then A is *complete for $\mathcal{C}$ (or $\mathcal{C}$-complete) under $\leq_m^{log}$*. Since we only use $\leq_m^{\log}$-reductions for decision problems, we just say A is hard for $\mathcal{C}$ (or $\mathcal{C}$-hard) or A is complete for $\mathcal{C}$ (or $\mathcal{C}$-complete).

It can be shown that $\leq_m^{\log}$ is transitive, i.e., $A \leq B$ and $B \leq D$ implies $A \leq D$ (see [Pap94]). Therefore if a problem A is hard for a class $\mathcal{C}$, then B is hard for $\mathcal{C}$ if and only if $A \leq_m^{\log} B$.

The class PSPACE and all classes from the polynomial hierarchy are closed under $\leq_m^{\log}$-reductions, i.e., if $\mathcal{C}$ is one of those classes and A and B are problems such that $B \in \mathcal{C}$, then $A \leq_m^{\log} B$ implies $A \in \mathcal{C}$.

We define the well known satisfiability problem, which is the most famous representative of all NP-complete problems:

Problem:	SAT
Input:	a propositional formula φ
Question:	does φ have a satisfying assignment?

In fact SAT was the first problem which was proven to be NP-complete in a seminal paper by Cook [Coo71].

Besides deciding whether a given instance satisfies some property there is a more general computational tasks: the problem of computing a function $f : \Sigma^* \to \Pi^*$ for some alphabets Σ and Π. These problems are called *function problems*. In Chapter 4 we study *counting problems*, which are a special case of function problems. A typical counting problem is the following generalization of SAT.

Problem:	#SAT
Input:	a propositional formula φ
Question:	how many satisfying assignments does φ have?

In general a counting problem is the task to compute a function $f : \Sigma^* \to \mathbb{N}$ for some alphabet Σ.

We define the complexity classes that are important in our study of counting problems. The complexity class FP is the class of all functions that can be computed by a deterministic Turing Machine in time bounded by a polynomial in the size of the input. The complexity class #P is the class of all counting functions f, such that there exists a non-deterministic Turing Machine, which stops in time bounded by a polynomial in the input size, and

which has on input w exactly $f(w)$ accepting computation paths. FP is the canonical generalization of the decision class P, and #P was introduced by Valiant [Val79b, Val79a]. It holds that every counting problem from FP also is in #P.

To relate the the complexity of counting problems we use *many-one counting reductions*:

Definition 2.1. Let $f : \Sigma^* \to \mathbb{N}$ and $g : \Pi^* \to \mathbb{N}$ be counting problems for some alphabets Σ and Π. Then f is *many-one counting reducible* (or simply *counting reducible*) to g in logarithmic space, if there exist functions $\alpha : \Sigma^* \to \Pi^*$ and $\beta : \mathbb{N} \to \mathbb{N}$, such that:

1. for every $x \in \Sigma^*$ holds: $f(x) = \beta(g(\alpha(x)))$, and
2. α and β can be computed by deterministic Turing machines in space bounded by $c \log n + d$, where n is the length of the input and $d, c \in \mathbb{N}$ are constants.

If β is the identity function, then f is *parsimonious reducible* to g in logarithmic space.

To denote that f reduces parsimonious to g in logarithmic space the notation $f \leq_!^{\log} g$ has been established. In this thesis we use additionally the notation $f \leq_c^{\log} g$ to indicate that f is counting reducible to g in logarithmic space.

We say a counting problem g is #P-hard under counting (parsimonious) reductions, if for every counting problem f from #P holds $f \leq_c^{\log} g$ ($f \leq_!^{\log} g$). The problem g is #P-complete under counting (parsimonious) reductions if it is from #P and additionally #P-hard under counting (parsimonious) reductions. If we only speak of #P-hardness and #P-completeness we mean #P-hardness and #P-completeness under counting reductions.

Again both reducibilities, $\leq_c^{\log}$ and $\leq_!^{\log}$, are transitive, therefore to show that some problem g is #P-hard under counting (parsimonious) reductions it is sufficient to show that there is some problem f that is #P-hard under counting (parsimonious) reductions, such that $f \leq_c^{\log} g$ ($f \leq_!^{\log} g$).

2.2 Relations and Constraints

In this section we will introduce constraint formulas which are the main subject of this thesis.

An n-ary relation R over a set D is a subset of D^n, in this case D is called the *domain* of R. In this thesis, if we speak of domains, we mean finite sets.

The *Boolean domain* is the set $\{0,1\}$. A *constraint language* over D is a set of non-empty relations over D.

We use different notions for relations. Sometimes we prefer to represent a relation $R \subseteq D^n$ as matrix by writing its tuples as row vectors in lexicographical order, or if D is the Boolean domain, we represent R as a propositional formula $\varphi(x_1, \ldots, x_n)$ with

$$R = \{(I(x_1), \ldots, I(x_n)) \mid I \text{ is a satisfying truth assignment for } \varphi)\}\,.$$

Example 2.2. 1. Let $R = \{(0,1,2),(1,2,0),(1,1,0)\}$ be a relation over the domain $\{0,1,2\}$. When representing R as matrix we write:

$$R = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 1 & 0 \\ 1 & 2 & 0 \end{pmatrix}$$

2. By $x_1 \vee \neg x_2$ we represent the relation $\{(0,0),(1,0),(1,1)\}$.

For a domain D and $a \in D$ we define the following relations over D:

$$\begin{aligned} \mathrm{C}_a &=_{def} \{(a)\} \\ \mathrm{Eq}_D &=_{def} \{(d,d) \mid d \in D\}\,. \end{aligned}$$

For the Boolean domain we define:

$$\begin{aligned} \mathrm{Eq} &=_{def} \mathrm{Eq}_{\{0,1\}} \\ \mathrm{Imp} &=_{def} \{0,1\}^2 \setminus \{(1,0)\} \\ \mathrm{Or}^k &=_{def} \{0,1\}^k \setminus \{(0,\ldots,0)\} \\ \text{1-in-3} &=_{def} \{(0,0,1),(0,1,0),(1,0,0)\} \\ \mathrm{Nae} &=_{def} \{0,1\}^3 \setminus \{(0,0,0),(1,1,1)\} \\ \mathrm{Dup} &=_{def} \{0,1\}^3 \setminus \{(0,1,0),(1,0,1)\} \\ \mathrm{Even}^k &=_{def} \left\{(a_1,\ldots,a_k) \in \{0,1\}^k \mid a_1 + \cdots + a_k \text{ is even}\right\} \\ \mathrm{Odd}^k &=_{def} \left\{(a_1,\ldots,a_k) \in \{0,1\}^k \mid a_1 + \cdots + a_k \text{ is odd}\right\}. \end{aligned}$$

Let D be a domain and let X be a set of variables. For an n-ary relation R over D and some variables $x_1, \ldots, n_n \in X$ an *R-clause* is of the form

$R(x_1, \ldots, x_n)$. We also call R-clauses *constraints*, or if R is from some constraint language Γ over D, we call them Γ-*clauses*. An *assignment* of D to X is a function $I : X \to D$. We say I *satisfies* the R-clause $R(x_1, \ldots, x_n)$ if $(I(x_1), \ldots, I(x_n)) \in R$.

A *constraint formula* over Γ (or a Γ-*formula*) is a finite conjunction of Γ-clauses, i.e., a formula of the form

$$\varphi = R_1(x_1^1, \ldots, x_{n_1}^1) \wedge \cdots \wedge R_k(x_1^k, \ldots, x_{n_k}^k),$$

where $R_i \in \Gamma$ is an n_i-ary relation and $x_j^i \in X$ for $1 \leq i \leq k$ and $1 \leq j \leq n_i$. By $\mathrm{Var}(\varphi) =_{def} \left\{x_j^i \mid 1 \leq i \leq k \text{ and } 1 \leq j \leq n_i\right\}$ we denote the set of variables appearing in φ. An assignment I of D to X *satisfies* φ if it satisfies every constraint in φ. An assignment of D to $\mathrm{Var}(\varphi)$ that satisfies φ is a *solution* for φ. If φ has a solution we say that φ is *satisfiable*. The set of all solutions for φ is denoted by $\mathrm{Sol}(\varphi)$.

Let Γ be a constraint language over some domain. A question very intensively studied in complexity theory is the *constraint satisfaction problem*:

Problem:	CSP(Γ)
Input:	a Γ-formula φ
Question:	is φ satisfiable?

Theorem 2.3 ([Sch78, Bul06])**.** *Let Γ be a finite constraint language over a domain with maximal three elements, then* CSP(Γ) *is complete for* NP*, or decidable in polynomial time.*

The dichotomy for two-element domains was proven in 1978 and is known as *Schaefer's Theorem*. The more recent result for constraint languages over three-element domains from Bulatov has been proven in 2002 and strengthens a conjecture by Feder and Vardi, saying that the constraint satisfaction problem is in P or NP-complete for all finite constraint languages over arbitrary finite domains [FV98].

2.3 Closure Properties

Two constraint formulas φ and ψ are equivalent ($\varphi \equiv \psi$) if they have exactly the same solutions. We define three different closures for constraint languages:

Definition 2.4. Let Γ be a constraint language over a domain D.

- $\langle\Gamma\rangle$ is the set of all relations R over D, such that $R = \emptyset$ or there exist distinct variables $x_1, \ldots, x_n, y_1, \ldots, y_k$ and a $\Gamma \cup \{\mathrm{Eq}_D\}$-formula φ such that

  $$R(x_1, \ldots, x_n) \text{ can be expressed by } \exists y_1 \ldots y_k \varphi.$$

  That means every solution of φ satisfies $R(x_1, \ldots, x_n)$ and every solution of $R(x_1, \ldots, x_n)$ can be extended to a solution of φ. We call $\langle\Gamma\rangle$ the *co-clone* generated by Γ and say that Γ is a *base* of $\langle\Gamma\rangle$.

- $\langle\Gamma\rangle_{\not\exists}$ is the set of all relations R over D, such that $R = \emptyset$ or there exist distinct variables $x_1, \ldots, x_n$ and a $\Gamma \cup \{\mathrm{Eq}_D\}$-formula φ such that

  $$R(x_1, \ldots, x_n) \text{ is equivalent to } \varphi.$$

  We call $\langle\Gamma\rangle_{\not\exists}$ the *weak system* generated by Γ and say that Γ is a *base* of $\langle\Gamma\rangle_{\not\exists}$.

- $\langle\Gamma\rangle_{\not\exists,\neq}$ is the set of all relations R over D, such that $R = \emptyset$ or there exist distinct variables $x_1, \ldots, x_n$ and a Γ-formula φ such that

  $$R(x_1, \ldots, x_n) \text{ is equivalent to } \varphi.$$

  We call $\langle\Gamma\rangle_{\not\exists,\neq}$ the *weak system without equality* generated by Γ and say that Γ is a *base* of $\langle\Gamma\rangle_{\not\exists,\neq}$.

Often co-clones are referred to as *relational clones* and weak systems (without equality) are sometimes called *weak systems with* 0 *(and without identification)*. Note that the $\neq$ in the notation $\langle . \rangle_{\not\exists,\neq}$, does not have the meaning of inequality, but symbolizes the disallowance of equality-clauses. In the case of one-element constraint languages $\Gamma = \{R\}$ we often write $\langle R\rangle$ instead of $\langle\{R\}\rangle$, etc.

It follows directly from the definitions that $\Gamma \subseteq \langle\Gamma\rangle_{\not\exists,\neq} \subseteq \langle\Gamma\rangle_{\not\exists} \subseteq \langle\Gamma\rangle$. Further the defined closures have the properties stated in the next proposition.

Proposition 2.5. *Let Γ_1, Γ_2 be constraint languages over a domain D and φ_1 a Γ_1-formula. Then the following holds:*

1. *$\Gamma_1 \subseteq \langle\Gamma_2\rangle$ implies that there exists a Γ_2-formula φ_2 that can be computed in logarithmic space from φ_1, and that is satisfiable if and only if φ_1 is;*

2. *$\Gamma_1 \subseteq \langle\Gamma_2\rangle_{\not\exists,\neq}$ implies that there exists a $\Gamma_2 \cup \{\mathrm{Eq}_D\}$-formula φ_2 that can be computed in logarithmic space from φ_1, and that is equivalent to φ_1;*

3. *$\Gamma_1 \subseteq \langle\Gamma_2\rangle_{\not\exists,\neq}$ implies that there exists a Γ_2-formula φ_2 that can be computed in logarithmic space from φ_1, and that is equivalent to φ_1.*

It follows immediately from part 1 of the previous proposition that, for finite constraint languages Γ_1 and Γ_2, it holds

$$\Gamma_1 \subseteq \langle \Gamma_2 \rangle \quad \text{implies} \quad \mathrm{CSP}(\Gamma_1) \leq_m^{\log} \mathrm{CSP}(\Gamma_2).$$

The following definitions lead to a strong connection between constraint languages and sets of functions.

Let D be a domain. A k-ary *partial D-valued function* is a function $f : A \to D$ with $A \subseteq D^k$ for a $k \geq 1$. We say f is *not defined* on $D^k \setminus A$. If $A = D^k$, then f is a *total D-valued function*. We denote theset of all total D-valued functions by OP_D. When speaking of D-valued functions we normally mean total D-valued functions. For the Boolean domain $D = \{0, 1\}$ we speak of *Boolean functions* and *partial Boolean functions*.

Definition 2.6. Let R be an n-ary relation over a domain D, let $A \subseteq D^k$ for some $k \geq 1$ and let $f : A \to D$ be a partial D-valued function. Then f is a *partial polymorphism* of R (or R is *invariant* under f) if for all $t_1, \ldots, t_k \in R$, such that $(t_1[i], \ldots, t_k[i]) \in A$ for every $i \in \{1, \ldots, n\}$, it holds

$$(\, f(t_1[1], \ldots, t_k[1]), \ldots, f(t_1[n], \ldots, t_k[n]) \,) \in R,$$

i.e., if R is closed under coordinate-wise application of f. If f is total, we say f is a *polymorphism* of R.

The set of all polymorphisms of a relation R is denoted by $\mathrm{Pol}(R)$, and the set of all partial polymorphisms by $\mathrm{pPol}(R)$. For a constraint language Γ, we say f is a (partial) polymorphism of Γ, if f is a (partial) polymorphism of each relation from Γ and we set $\mathrm{Pol}(\Gamma) =_{def} \bigcap_{R \in \Gamma} \mathrm{Pol}(R)$ and $\mathrm{pPol}(\Gamma) =_{def} \bigcap_{R \in \Gamma} \mathrm{pPol}(R)$.

For a partial D-valued function f, we denote the set of all relations over D that are invariant under f by $\mathrm{Inv}(f)$. Accordingly the set of all relations invariant under a set $\mathcal{F}$ of partial D-valued functions, $\mathrm{Inv}(\mathcal{F}) = \bigcap_{f \in \mathcal{F}} \mathrm{Inv}(f)$, consists of all relations that are invariant under all functions from $\mathcal{F}$.

We often apply functions coordinate wise to tuples, like in Definition 2.6. For an n-ary D-valued partial function and tuples $t_1, \ldots, t_n \in D^k$ we set

$$f(t_1, \ldots, t_n) =_{def} (\, f(t_1[1], \ldots, t_k[1]), \ldots, f(t_1[n], \ldots, t_k[n]) \,)$$

to simplify notation.

We define some D-valued functions: the function $pr_i^k : D^k \to D$, defined by $pr_i^k(a_1, \ldots, a_k) = a_i$ for every $(a_1, \ldots, a_i) \in D^k$, is the k-ary *projection* to the i-th component; $id =_{def} pr_1^1$ is the identity; for $a \in D$ the *constant* function $c_a : D \to D$, defined by $c_a(d) = a$ for all $d \in D$.

Let $f, g_1, \ldots, g_n$ be partial D-valued functions such that f is of arity n and $g_1, \ldots, g_n$ are of arity k. Then the *composition* of f and $(g_1, \ldots, g_n)$ is the k-ary partial D-valued function

$$f \circ (g_1, \ldots, g_n) : A \to D,$$

defined on

$$A = \big\{ t \in D^k \,|\, g_1, \ldots, g_n \text{ are defined on } t \text{ and } f \text{ is defined on } (g_1(t), \ldots, g_n(t)) \big\},$$

such that for every $(a_1, \ldots, a_k) \in A$ holds

$$f \circ (g_1, \ldots, g_n)(a_1, \ldots, a_k) = f(g_1(a_1, \ldots, a_k), \ldots, g_n(a_1, \ldots, a_k)).$$

For a function $f : A \to B$ and a subset A' of A, the function $f|_{A'} : A' \to B$ defined by $f|_{A'}(a) = f(a)$ for every $a \in A'$ is called the *restriction* of f to A'.

Definition 2.7. Let D be a domain.

1. A set of total D-valued functions, that contains all projections and is closed under arbitrary composition, is called a *clone* over D. For a set $\mathcal{F}$ of total D-valued functions, $[\mathcal{F}]$ denotes the smallest clone containing $\mathcal{F}$.

2. A set of partial D-valued functions that contains all projections, is closed under arbitrary composition, and is closed under restriction of functions, is called a *strong partial clone* over D. For a set $\mathcal{F}$ of partial D-valued functions, $[\mathcal{F}]_p$ denotes the smallest strong partial clone containing $\mathcal{F}$.

It holds for every constraint language Γ that $\mathrm{Pol}(\Gamma)$ is a clone and $\mathrm{pPol}(\Gamma)$ is a strong partial clone. Conversely, $\mathrm{Inv}(\mathcal{F})$ forms a co-clone if $\mathcal{F}$ is a set of total D-valued functions, and a weak system if $\mathcal{F}$ is a set of partial D-valued functions.

We say a set of D-valued (partial D-valued) functions $\mathcal{F}$ is a *base* of a clone (strong partial clone) $\mathcal{C}$ if $[\mathcal{F}] = \mathcal{C}$ ($[\mathcal{F}]_p = \mathcal{C}$).

The next two results show very strong connections between clones and co-clones and between strong partial clones and weak systems. The Galois correspondences stated in Proposition 2.8 follow directly from the definitions of the operators $\mathrm{Pol}, \mathrm{pPol}$ and Inv.

Proposition 2.8. *Let D be a domain.*

1. Pol *and* Inv *form a Galois correspondence between the set of all D-valued functions and the set of all relations over D, i.e., for sets of relations*

Γ_1, Γ_2 *over* D *and sets of* D*-valued functions* $\mathcal{F}_1, \mathcal{F}_2$ *it holds that*

$$\Gamma_1 \subseteq \Gamma_2 \Rightarrow \mathrm{Pol}(\Gamma_1) \supseteq \mathrm{Pol}(\Gamma_2),$$
$$\mathcal{F}_1 \subseteq \mathcal{F}_2 \Rightarrow \mathrm{Inv}(\mathcal{F}_1) \supseteq \mathrm{Inv}(\mathcal{F}_2),$$
$$\Gamma_1 \subseteq \mathrm{Inv}(\mathrm{Pol}(\Gamma_1)),$$
$$\mathcal{F}_1 \subseteq \mathrm{Pol}(\mathrm{Inv}(\mathcal{F}_1)).$$

2. pPol *and* Inv *form a Galois correspondence between the set of all partial* D*-valued functions and the set of all relations over* D*, i.e., for sets of relations* Γ_1, Γ_2 *over* D *and sets of partial* D*-valued functions* $\mathcal{F}_1, \mathcal{F}_2$ *it holds that*

$$\Gamma_1 \subseteq \Gamma_2 \Rightarrow \mathrm{pPol}(\Gamma_1) \supseteq \mathrm{pPol}(\Gamma_2),$$
$$\mathcal{F}_1 \subseteq \mathcal{F}_2 \Rightarrow \mathrm{Inv}(\mathcal{F}_1) \supseteq \mathrm{Inv}(\mathcal{F}_2),$$
$$\Gamma_1 \subseteq \mathrm{Inv}(\mathrm{pPol}(\Gamma_1)),$$
$$\mathcal{F}_1 \subseteq \mathrm{pPol}(\mathrm{Inv}(\mathcal{F}_1)).$$

Points *1a* and *1b* in the next theorem were proven independently by Geiger [Gei68], and by Bodnarchuk, Kalužnin, Kotov and Romov [BKKR69]. Points *2a* and *2b* are from Romov [Rom81], but an implicit proof can be found in [Gei68] as well.

Theorem 2.9 ([Gei68, BKKR69, Rom81])**.** *Let* D *be a domain,* Γ *a constraint language over* D*,* $\mathcal{F}$ *a set of* D*-valued functions, and* $\mathcal{B}$ *a set of partial* D*-valued functions. The following equations hold:*

1a. $\mathrm{Inv}(\mathrm{Pol}(\Gamma)) = \langle\Gamma\rangle$

1b. $\mathrm{Pol}(\mathrm{Inv}(\mathcal{F})) = [\mathcal{F}]$

2a. $\mathrm{Inv}(\mathrm{pPol}(\Gamma)) = \langle\Gamma\rangle_{\not\equiv}$

2b. $\mathrm{pPol}(\mathrm{Inv}(\mathcal{B})) = [\mathcal{B}]_p$

It follows from the previous results that there is a one-to-one correspondence between clones and co-clones and a one-to-one correspondence between strong partial clones and weak systems. Each clone $\mathcal{C}$ corresponds uniquely to the co-clone $\mathrm{Inv}(\mathcal{C})$ and each strong partial clone $\mathcal{D}$ to the weak system $\mathrm{Inv}(\mathcal{D})$. The next corollary is a list of simple conclusions from the previous theorems.

Corollary 2.10. *Let* D *be a domain,* Γ_1 *and* Γ_2 *constraint languages over* D*,* $\mathcal{F}_1$ *and* $\mathcal{F}_2$ *sets of* D*-valued functions and* $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *sets of partial* D*-valued functions.*

1a. $\mathrm{Pol}(\Gamma_1) = \mathrm{Pol}(\Gamma_2)$ *if and only if* $\langle\Gamma_1\rangle = \langle\Gamma_2\rangle$.

1b. $\mathrm{Inv}(\mathcal{F}_1) = \mathrm{Inv}(\mathcal{F}_2)$ *if and only if* $[\mathcal{F}_1] = [\mathcal{F}_2]$.

2a. $\mathrm{pPol}(\Gamma_1) = \mathrm{pPol}(\Gamma_2)$ *if and only if* $\langle\Gamma_1\rangle_{\not\exists} = \langle\Gamma_2\rangle_{\not\exists}$.

2b. $\mathrm{Inv}(\mathcal{B}_1) = \mathrm{Inv}(\mathcal{B}_2)$ *if and only if* $[\mathcal{B}_1]_p = [\mathcal{B}_2]_p$.

For each domain D the inclusion structure of all clones over D forms a lattice, as well as the inclusion structures of all strong partial clones, co-clones and weak systems over D. It holds that the lattice of all clones over D is dual to the lattice of all co-clones D and the lattice of all strong partial clones over D is dual to the lattice of weak systems over D.

The lattice of all clones over the Boolean domain is well known: in [Pos41] Post identified all Boolean clones and found a finite base for each of them. In Table 2.3 all Boolean clones and Post's bases are listed.

We often express Boolean functions as propositional formulas: a propositional formula $\varphi(x_1, \ldots, x_n)$ represents the n-ary function f, such that for all $a_1, \ldots, a_n \in \{0, 1\}$ it holds $f(a_1, \ldots, a_n) = 1$ if and only if there is a satisfying assignment I for φ such that $I(x_1) = a_1, \ldots, I(x_n) = a_n$. For example $x \vee y$ represents the function that gives 1 if and only if at least one of its arguments are 1. Sometimes we denote Boolean functions by Boolean operators, for instance we write only $\vee$ instead of $x \vee y$, $\rightarrow$ instead of $x \rightarrow y$, or $\neg$ instead of $\neg(x)$ or $\overline{x}$. The Boolean constant functions c_0 and c_1 we denote sometimes with 0 and 1.

For $n \geq 1$ we define $h_n : D^{n+1} \rightarrow D$ to be the $n+1$-ary function that gives 1 if and only if at least n of its arguments are 1, i.e., $h_n(a_1, \ldots, a_{n+1}) = 1$ if and only if $a_1 + \cdots + a_{n+1} \geq n$.

In the following we define some properties of Boolean functions. Let $f : \{0, 1\}^k \rightarrow \{0, 1\}$ a k-ary Boolean function. For $a \in \{0, 1\}$ we say f is *a-reproducing*, if $f(a, \ldots, a) = a$. We say f is a-separating if there exists an $i \in \{1, \ldots, k\}$ such that for every $t \in D^k$ with $f(t) = a$ it holds that $t[i] = a$. For some $m \geq 2$ we call f a-separating of degree m, if for every $A \subseteq D^k$, such that $|A| = m$ and $f(t) = a$ for all $t \in A$, there exists an $i \in \{1, \ldots, k\}$ such that $t[i] = a$ for all $t \in A$. We say f is *monotone* if $a_1 \leq b_1, \ldots, a_k \leq b_k$ implies $f(a_1, \ldots, a_k) \leq f(b_1, \ldots, b_k)$. The k-ary Boolean function dual(f) is defined by $\mathrm{dual}(f)(a_1, \ldots, a_k) = \overline{f(\overline{a_1}, \ldots, \overline{a_n})}$. If $f = \mathrm{dual}(f)$, then f is *self-dual*. Finally f is *linear* if there is a set of indices $I \subseteq \{1, \ldots, k\}$, such that either for all $a_1, \ldots, a_k \in \{0, 1\}$ it holds $f(a_1, \ldots, a_k) = 1$ if and only if $\sum_{i \in I} a_i$ is even, or for all $a_1, \ldots, a_k \in \{0, 1\}$ it holds $f(a_1, \ldots, a_k) = 1$ if and only if $\sum_{i \in I} a_i$ is odd.

We denote the co-clone that corresponds to a Boolean clone $\mathcal{C}$ by $\mathrm{I}\mathcal{C} =_{def} \mathrm{Inv}(\mathcal{C})$. Figure 2.3 shows the lattice of all Boolean co-clones which is dual to Post's lattice of all Boolean clones. A list of finite or, in the not finitely generated cases, uniform bases for the Boolean co-clones was was presented by Böler et al. [BRSV05] and can be seen in Table 3.2 in Chapter 3.

For an n-ary Boolean relation R, we define

$$\mathrm{dual}(R) =_{def} \{(\overline{a_1}, \ldots, \overline{a_n}) \in \{0,1\}^n \mid (a_1, \ldots, a_n) \in R\}$$

to be the relation *dual* to R. For a set of relations Γ, we set $\mathrm{dual}(\Gamma) =_{def} \{\mathrm{dual}(R) \mid R \in \Gamma\}$. Note that $\mathrm{dual}(\mathrm{dual}(\Gamma)) = \Gamma$ and that a Boolean function f is is a polymorphism of Γ if and only if $\mathrm{dual}(f)$ is a polymorphism of $\mathrm{dual}(\Gamma)$. The dual of a Boolean co-clone can be found in Figure 2.3 via the vertical symmetry axis: it is the mirror-image of the original co-clone. For example $\mathrm{dual}(\mathrm{IV}_1) = \mathrm{IE}_0$.

We now can state Schaefer's Theorem in more detail.

Theorem 2.11 ([Sch78])**.** *Let* Γ *be a finite constraint language over* $\{0,1\}$*. Then* $\mathrm{CSP}(\Gamma)$ *is* NP*-complete, if* $\mathrm{IN}_2 \subseteq \langle\Gamma\rangle$*. Otherwise it holds* $\mathrm{CSP}(\Gamma) \in \mathrm{P}$*.*

The counting version of the constraint satisfiability problem is defined as:

Problem:	$\#\mathrm{CSP}(\Gamma)$
Input:	a Γ-formula φ
Question:	how many solutions does φ have?

Creignou and Hermann classified the complexity of $\#\mathrm{CSP}(\Gamma)$ for every finite Boolean constraint language Γ.

Theorem 2.12 ([CH96])**.** *Let* Γ *be a finite constraint language over* $\{0,1\}$*. If* $\langle\Gamma\rangle \subseteq \mathrm{IL}_2$*, then* $\mathrm{CSP}(\Gamma) \in \mathrm{FP}$*. Otherwise* $\mathrm{CSP}(\Gamma)$ *is* #P*-complete.*

We say a Boolean constraint language Γ is *Schaefer*, if $\mathrm{IN} \not\subseteq \langle\Gamma\rangle$. That means Γ is Schaefer if and only if Γ is a subset of one of the following co-clones: IV_2, IE_2, ID_2, IL_2. For many problems in the constraint context the Schaefer property guarantees efficient solvability.

Clone	**Definition**	**Base**
BF	All Boolean functions	$\{\vee, \wedge, \neg\}$
R_0	$\{f \in \mathrm{BF} \mid f \text{ is 0-reproducing }\}$	$\{\wedge, \oplus\}$
R_1	$\{f \in \mathrm{BF} \mid f \text{ is 1-reproducing }\}$	$\{\vee, \leftrightarrow\}$
R_2	$\mathrm{R}_1 \cap \mathrm{R}_0$	$\{\vee, x \wedge (y \leftrightarrow z)\}$
M	$\{f \in \mathrm{BF} \mid f \text{ is monotone }\}$	$\{\vee, \wedge, 0, 1\}$
M_0	$\mathrm{M} \cap \mathrm{R}_0$	$\{\vee, \wedge, 0\}$
M_1	$\mathrm{M} \cap \mathrm{R}_1$	$\{\vee, \wedge, 1\}$
M_2	$\mathrm{M} \cap \mathrm{R}_2$	$\{\vee, \wedge\}$
S_0^m	$\{f \in \mathrm{BF} \mid f \text{ is 0-separating of degree } m\}$	$\{\rightarrow, \mathrm{dual}(h_m)\}$
S_0	$\{f \in \mathrm{BF} \mid f \text{ is 0-separating }\}$	$\{\rightarrow\}$
S_{02}^m	$\mathrm{S}_0^m \cap \mathrm{R}_2$	$\{x \vee (y \wedge \overline{z}), \mathrm{dual}(h_m)\}$
S_{02}	$\mathrm{S}_0 \cap \mathrm{R}_2$	$\{x \vee (y \wedge \overline{z})\}$
S_{01}^m	$\mathrm{S}_0^m \cap \mathrm{M}$	$\{\mathrm{dual}(h_m), 1\}$
S_{01}	$\mathrm{S}_0 \cap \mathrm{M}$	$\{x \vee (y \wedge z), 1\}$
S_{00}^m	$\mathrm{S}_0^m \cap \mathrm{R}_2 \cap \mathrm{M}$	$\{x \vee (y \wedge z), \mathrm{dual}(h_m)\}$
S_{00}	$\mathrm{S}_0 \cap \mathrm{R}_2 \cap \mathrm{M}$	$\{x \vee (y \wedge z)\}$
S_1^m	$\{f \in \mathrm{BF} \mid f \text{ is 1-separating of degree } m\}$	$\{x \wedge \overline{y}, h_m\}$
S_1	$\{f \in \mathrm{BF} \mid f \text{ is 1-separating }\}$	$\{x \wedge \overline{y}\}$
S_{12}^m	$\mathrm{S}_1^m \cap \mathrm{R}_2$	$\{x \wedge (y \vee \overline{z}), h_m\}$
S_{12}	$\mathrm{S}_1 \cap \mathrm{R}_2$	$\{x \wedge (y \vee \overline{z})\}$
S_{11}^m	$\mathrm{S}_1^m \cap \mathrm{M}$	$\{h_m, 0\}$
S_{11}	$\mathrm{S}_1 \cap \mathrm{M}$	$\{x \wedge (y \vee z), 0\}$
S_{10}^m	$\mathrm{S}_1^m \cap \mathrm{R}_2 \cap \mathrm{M}$	$\{x \wedge (y \vee z), h_m\}$
S_{10}	$\mathrm{S}_1 \cap \mathrm{R}_2 \cap \mathrm{M}$	$\{x \wedge (y \vee z)\}$
D	$\{f \in \mathrm{BF} \mid f \text{ is self-dual}\}$	$\{x\overline{y} \vee x\overline{z} \vee (\overline{y} \wedge \overline{z})\}$
D_1	$\mathrm{D} \cap \mathrm{R}_2$	$\{xy \vee x\overline{z} \vee y\overline{z}\}$
D_2	$\mathrm{D} \cap \mathrm{M}$	$\{xy \vee yz \vee xz\}$
L	$\{f \in \mathrm{BF} \mid f \text{ is linear}\}$	$\{\oplus, 1\}$
L_0	$\mathrm{L} \cap \mathrm{R}_0$	$\{\oplus\}$
L_1	$\mathrm{L} \cap \mathrm{R}_1$	$\{\leftrightarrow\}$
L_2	$\mathrm{L} \cap \mathrm{R}$	$\{x \oplus y \oplus z\}$
L_3	$\mathrm{L} \cap \mathrm{D}$	$\{x \oplus y \oplus z \oplus 1\}$
V	$\{f \in \mathrm{BF} \mid f \text{ is constant or an } n-\text{ary OR function}\}$	$\{\vee, 0, 1\}$
V_0	$[\{\vee\}] \cup [\{0\}]$	$\{\vee, 0\}$
V_1	$[\{\vee\}] \cup [\{1\}]$	$\{\vee, 1\}$
V_2	$[\{\vee\}]$	$\{\vee\}$
E	$\{f \in \mathrm{BF} \mid f \text{ is constant or an } n-\text{ary AND function}\}$	$\{\wedge, 0, 1\}$
E_0	$[\{\wedge\}] \cup [\{0\}]$	$\{\wedge, 0\}$
E_1	$[\{\wedge\}] \cup [\{1\}]$	$\{\wedge, 1\}$
E_2	$[\{\wedge\}]$	$\{\wedge\}$
N	$[\{\neg\}] \cup [\{0\}] \cup [\{1\}]$	$\{\neg, 1\}$
N_2	$[\{\neg\}]$	$\{\neg\}$
I	$[\{0\}] \cup [\{1\}]$	$\{id, 0, 1\}$
I_0	$[\{0\}]$	$\{id, 0\}$
I_1	$[\{1\}]$	$\{id, 1\}$
I_2	$[\{id\}]$	$\{id\}$

Table 2.1: Bases for all Boolean Clones

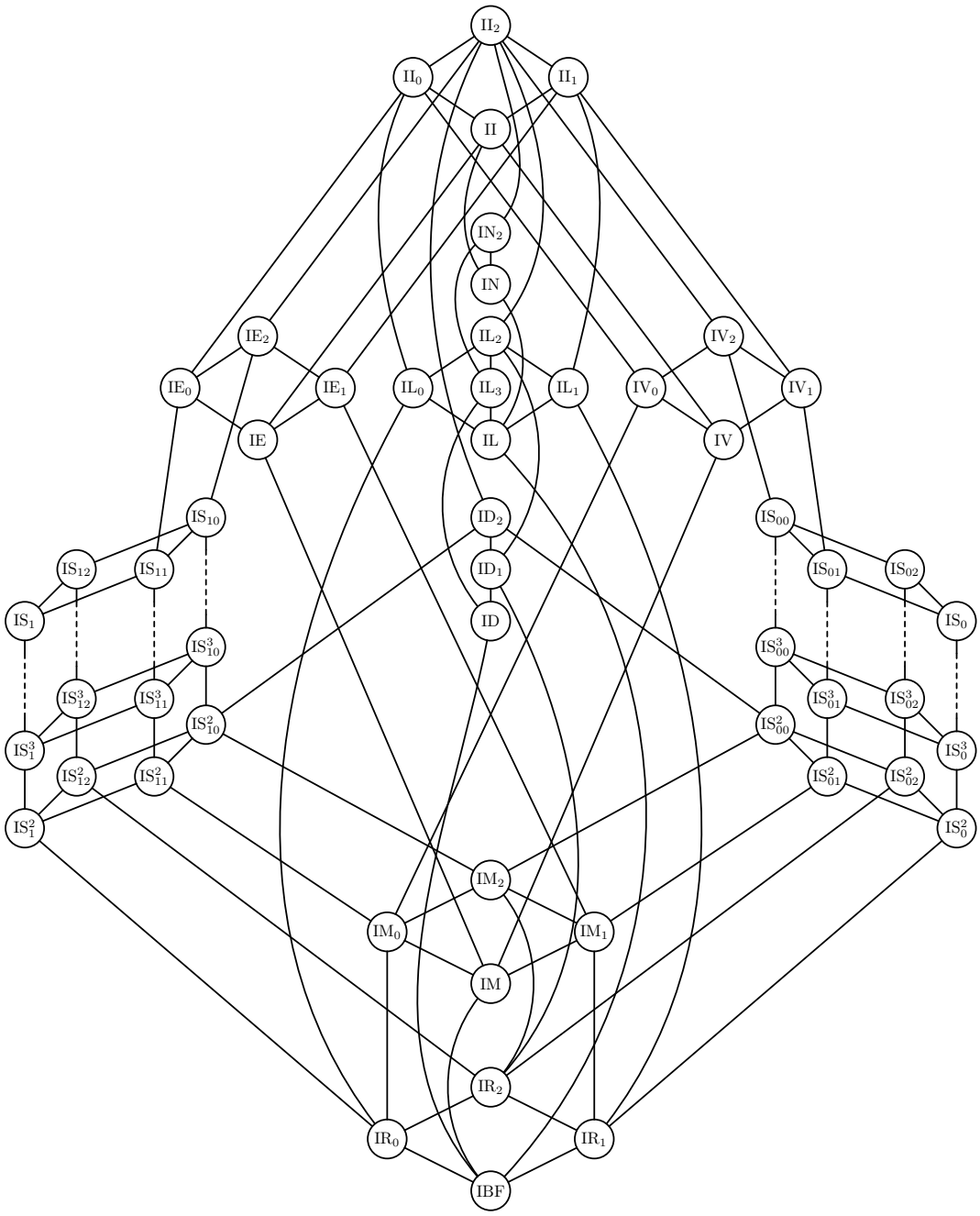

Figure 2.2: The Lattice of all Boolean Co-Clones

Chapter 3

Weak Bases

The knowledge about the Boolean clones provided by Post in his seminal paper [Pos41], has become an important tool in the constraint satisfaction context. There is an easy proof for Schaefer's Theorem (see Theorem 2.11) that uses Post's characterization of the Boolean clones given in Table 2.3 and the strong connection between clones and co-clones stated in Proposition 2.8 and Theorem 2.9.

The reason why we can rely on this connection when classifying the complexity of CSP(Γ) for each finite constraint language Γ is that the complexity of CSP(Γ) depends only on the co-clone generated by Γ. Proposition 2.5 implies that, if two finite constraint languages Γ_1 and Γ_2 generate the same co-clone, then Γ_1-formulas can be transformed in satisfiability equivalent Γ_2-formulas and vice versa.

However, there are problems in the constraint satisfaction context for which this transformation does not give a reduction. For example for non-Boolean domains it is not clear whether the complexity of the problem EQUIV(Γ), which is the question if two given Γ-formulas are equivalent, depends only on the co-clone $\langle\Gamma\rangle$. Similarly CSP*(Γ), the problem whether a given Γ-formula has a non-constant solution, is not known to be complexity-invariant under the property of generating the same co-clone. In both cases the existential quantifiers appearing in the definition of the co-clone closure are the obstacle.

It is obvious that the weak systems without equality are a more suitable structure when looking at the equivalence problem, because the $\langle . \rangle_{\not\equiv,\neq}$-closure allows to transform a Γ_1-formula in an equivalent Γ_2-formula, if Γ_1 and Γ_2 generate the same weak system without equality (see Proposition 2.5). For CSP*(Γ) it can be shown that the complexity depends only on the weak system generated by Γ. Since weak systems and strong partial clones correspond to each other in the same way as clones and co-clones do, one can use the theory of strong partial clones to classify the complexity of CSP*(Γ). But there is

a big drawback: the strong partial clones are not fully identified, not even in the Boolean case. An aggravating factor is that the lattice of all Boolean strong partial clones, and so the lattice of all Boolean weak systems, contains an uncountable number of classes, whereas the number of Boolean clones is countable.

In this chapter we identify some strong partial clones, which play a special role for many classifications in the constraint satisfaction context. Every clone $\mathcal{C}$ canonically corresponds to a set of strong partial clones. We will detect and characterize the largest strong partial clone in this set. The intuition is that this strong partial clone corresponds to the "easiest" weak system whose polymorphisms are exactly $\mathcal{C}$, therefore lower complexity bounds for this "easy" weak system, transfer to all weak systems corresponding to $\mathcal{C}$.

We show how to construct bases, which we call *weak bases*, for these kind of weak systems and give for every Boolean co-clone a weak base that is minimal in a certain sense. We also give a method to identify weak bases that generate the "easiest" weak system without equality corresponding to a certain co-clone.

Later, in Chapters 4 and 5, we will see how the theory of weak bases can be used as a tool to classify the complexity of problems in the constraint context.

3.1 Small Weak Systems

Since strong partial clones over D are closed under arbitrary composition and contain all projections over D, their total functions form clones. For a clone $\mathcal{C}$ we let

$$\mathcal{I}(\mathcal{C}) =_{def} \{B \mid B \text{ strong partial clone such that } B \cap \mathrm{OP}_D = \mathcal{C}\}$$

denote the set of all strong partial clones that, reduced to only their total functions, are equal to $\mathcal{C}$. The notation $\mathcal{I}(\mathcal{C})$ stems from the fact that $\mathcal{I}(\mathcal{C})$ is an interval in the lattice of all strong partial clones over D.

Clearly $[\mathcal{C}]_p$ is the smallest strong partial clone in $\mathcal{I}(\mathcal{C})$. We will show that $\mathcal{I}(\mathcal{C})$ also has a largest element.

Let

$$\mathcal{I}_{\cup}(\mathcal{C}) =_{def} \bigcup_{B \in \mathcal{I}(\mathcal{C})} B$$

be the union of all partial clones whose total functions are exactly $\mathcal{C}$. We will give a characterization of $\mathcal{I}_{\cup}(\mathcal{C})$ and show that it again is a strong partial clone from $\mathcal{I}(\mathcal{C})$.

We need the following definition to characterize $\mathcal{I}_{\cup}(\mathcal{C})$.

Definition 3.1. Let $\mathcal{C}$ be a clone over a domain D and $f : E \to D$ an n-ary partial D-valued function. Then f is $\mathcal{C}$-total, if for all $m > 0$ and for all m-ary functions $g_1, \ldots, g_n \in \mathcal{C}$ the composition $f \circ (g_1, \ldots, g_n)$ is either non-total or from $\mathcal{C}$.

That means in other words that f is $\mathcal{C}$-total if and only if $[\mathcal{C} \cup \{f\}]_p \in \mathcal{I}(\mathcal{C})$. The next theorem shows that $\mathcal{C}$-totality is preserved by all operations involved in generating strong partial clones, that is by arbitrary composition and restriction.

Theorem 3.2. *Let $\mathcal{C}$ be a clone over a domain D and $\mathcal{F}$ a set of $\mathcal{C}$-total functions. Then the following holds:*

1. *every function from $[\mathcal{F}]_p$ is $\mathcal{C}$-total,*
2. *the set of all $\mathcal{C}$-total functions forms a strong partial clone.*

Proof. 1. Since obviously all projections over D are $\mathcal{C}$-total, it is sufficient to prove that arbitrary composition and restriction of functions preserve $\mathcal{C}$-totality.

Let $f, g_1, \ldots, g_n$ be $\mathcal{C}$-total partial D-valued functions such that f is of arity n and $g_1, \ldots, g_n$ are of arity m for an $m > 0$. Let $h_1, \ldots, h_m \in \mathcal{C}$ be k-ary functions. Then it holds:

$$(f \circ (g_1, \ldots, g_n)) \circ (h_1, \ldots, h_m) = f \circ (g_1 \circ (h_1, \ldots, h_m), \ldots, g_n \circ (h_1, \ldots, h_m)).$$

We distinguish two cases:

Case 1: $g_i \circ (h_1, \ldots, h_m) \in \mathcal{C}$ for all $i \in \{1, \ldots, n\}$. Then it holds that $(f \circ (g_1, \ldots, g_n)) \circ (h_1, \ldots, h_m)$ is either from $\mathcal{C}$ or non-total, since f is $\mathcal{C}$-total.

Case 2: there is an $i \in \{1, \ldots, n\}$ such that $g_i \circ (h_1, \ldots, h_m) \notin \mathcal{C}$. Since g_i is $\mathcal{C}$-total, it holds that the composition $g_i \circ (h_1, \ldots, h_m)$ is non-total. If $g_i \circ (h_1, \ldots, h_m)$ is not defined for some $v \in D^k$, then $(f \circ (g_1, \ldots, g_n)) \circ (h_1, \ldots, h_m)$ is not defined for v either. Therefore $(f \circ (g_1, \ldots, g_n)) \circ (h_1, \ldots, h_m)$ is non-total as well.

Thus $f \circ (g_1, \ldots, g_n)$ is $\mathcal{C}$-total.

Now let g_1' be a restriction of g_1. If $g_1' \circ (h_1, \ldots, h_m)$ is total, then it is equal to $g_1 \circ (h_1, \ldots, h_m)$. Since g_1 is $\mathcal{C}$-total, g_1' also is $\mathcal{C}$-total. It follows that $[\mathcal{F}]_p$ contains only $\mathcal{C}$-total functions.

2. It follows directly from the above that the set of of all $\mathcal{C}$-total functions is closed under arbitrary composition and restriction, therefore it forms a strong partial clone. □

The following theorem shows that $\mathcal{I}_{\cup}(\mathcal{C})$ consists of all $\mathcal{C}$-total functions. From this result we conclude that $\mathcal{I}(\mathcal{C})$ indeed has a greatest element which is the strong partial clone of all $\mathcal{C}$-total functions.

Theorem 3.3. *Let $\mathcal{C}$ be a clone over the domain D. Then $\mathcal{I}_{\cup}(\mathcal{C})$ is exactly the strong partial clone of all $\mathcal{C}$-total functions.*

Proof. We start by proving that every function from $\mathcal{I}_{\cup}(\mathcal{C})$ is $\mathcal{C}$-total. Let f be from $\mathcal{I}_{\cup}(\mathcal{C})$ with arity n. That means there exists a strong partial clone $\mathcal{B} \in \mathcal{I}(\mathcal{C})$ such that $f \in \mathcal{B}$. Assume f is not $\mathcal{C}$-total. Then there are $g_1, \ldots, g_n \in \mathcal{C}$, such that $h =_{def} f \circ (g_1, \ldots, g_n) \in \mathrm{OP}_D \setminus \mathcal{C}$. Since $\mathcal{B} \in \mathcal{I}(\mathcal{C})$, it holds that $\mathcal{C} \subseteq \mathcal{B}$, and this implies $h \in \mathcal{B}$, which is a contradiction to $h \in \mathrm{OP}_D \setminus \mathcal{C}$. Hence, f is $\mathcal{C}$-total.

Now we show that every $\mathcal{C}$-total function is from $\mathcal{I}_{\cup}(\mathcal{C})$. Let f be a $\mathcal{C}$-total function. Since obviously every function from $\mathcal{C}$ is $\mathcal{C}$-total, it follows from Theorem 3.2 that every function from $[\{f\} \cup \mathcal{C}]_p$ is $\mathcal{C}$-total. None of the total functions from $\mathrm{OP}_D \setminus \mathcal{C}$ is $\mathcal{C}$-total, therefore $[\{f\} \cup \mathcal{C}]_p \cap \mathrm{OP}_D = \mathcal{C}$. This means $[\{f\} \cup \mathcal{C}]_p \in \mathcal{I}(\mathcal{C})$, thus $f \in \mathcal{I}_{\cup}(\mathcal{C})$. □

So we know that $\mathcal{I}_{\cup}(\mathcal{C})$ is the greatest strong partial clone in $\mathcal{I}(\mathcal{C})$. We are now interested in the associated weak system $\mathrm{Inv}(\mathcal{I}_{\cup}(\mathcal{C}))$. The following definition is central to our approach of using strong partial clones to obtain complexity classifications.

Definition 3.4. Let $\mathcal{C}$ be a clone over a domain D and let Γ be a constraint language over D. If $\langle\Gamma\rangle_{\not\exists} = \mathrm{Inv}(\mathcal{I}_{\cup}(\mathcal{C}))$, we call Γ a *weak base* for the co-clone $\mathrm{I}\mathcal{C}$.

The name *weak base* is motivated by the next corollary which shows that a weak base Γ for a co-clone $\mathrm{I}\mathcal{C}$ is a co-clone base of $\mathrm{I}\mathcal{C}$ and generates the smallest weak system of all co-clone bases of $\mathrm{I}\mathcal{C}$.

Corollary 3.5. *Let $\mathcal{C}$ be a clone over D and Γ a weak base for $\mathrm{I}\mathcal{C}$. Then the following holds:*

1. $\langle\Gamma\rangle_{\not\exists} = \mathrm{Inv}(\mathcal{I}_{\cup}(\mathcal{C}))$
2. $\langle\Gamma\rangle = \mathrm{I}\mathcal{C}$
3. *If $\langle\Gamma\rangle = \langle\Gamma'\rangle$ for some constraint language Γ' over D, then $\langle\Gamma\rangle_{\not\exists} \subseteq \langle\Gamma'\rangle_{\not\exists}$*

Proof. 1. Since $\mathrm{pPol}(\Gamma) = \mathcal{I}_\cup(\mathcal{C})$, it holds $\mathrm{Inv}(\mathrm{pPol}(\Gamma)) = \mathrm{Inv}(\mathcal{I}_\cup(\mathcal{C}))$. Due to Theorem 2.9 it follows that $\mathrm{Inv}(\mathrm{pPol}(\Gamma)) = \langle\Gamma\rangle_{\not\equiv}$, which proves this point.

2. It holds that $\mathrm{Pol}(\Gamma) = \mathcal{I}_\cup(\mathcal{C}) \cap \mathrm{OP}_D = \mathcal{C}$, since $\mathrm{pPol}(\Gamma) = \mathcal{I}_\cup(\mathcal{C})$. Because of Theorem 2.9, it follows $\mathrm{Inv}(\mathrm{Pol}(\Gamma)) = \langle\Gamma\rangle$, therefore $\langle\Gamma\rangle = \mathrm{I}\mathcal{C}$.

3. Let Γ' be a constraint language over D such that $\langle\Gamma'\rangle = \langle\Gamma\rangle$. With Corollary 2.10 it follows that $\mathrm{Pol}(\Gamma') = \mathrm{Pol}(\Gamma) = \mathcal{C}$. Then $\mathrm{pPol}(\Gamma') \in \mathcal{I}(\mathcal{C})$ and therefore $\mathrm{pPol}(\Gamma') \subseteq \mathcal{I}_\cup(\mathcal{C}) = \mathrm{pPol}(\Gamma)$. Since pPol and Inv form a Galois correspondence as stated in Proposition 2.8, it holds $\mathrm{Inv}(\mathrm{pPol}(\Gamma)) \subseteq \mathrm{Inv}(\mathrm{pPol}(\Gamma'))$. Using Theorem 2.9 we conclude that $\langle\Gamma\rangle_{\not\equiv} \subseteq \langle\Gamma'\rangle_{\not\equiv}$. □

This shows that $\mathrm{Inv}(\mathcal{I}_\cup(\mathcal{C}))$ is the smallest weak system that is included in $\mathrm{I}\mathcal{C}$ but not in any co-clone which is a proper subset of $\mathrm{I}\mathcal{C}$.

Corollary 3.5 says that the relations from $\mathrm{Inv}(\mathcal{I}_\cup(\mathcal{C}))$ can be expressed by constraint formulas over $\Gamma \cup \{\mathrm{Eq}_D\}$ for every constraint language Γ such that $\mathrm{Pol}(\Gamma) = \mathcal{C}$. This property is essential for the complexity classifications we show in Chapters 4 and 5.

The rest of this chapter is devoted to finding finite weak bases for $\mathrm{I}\mathcal{C}$. We need the following definitions to construct weak bases: let R be a relation. $R(l, k)$ is the value at row l and column k in the matrix representation of R. Note that this is a unique notation because the rows in the matrix representation of R are ordered lexicographically. By $R(l, -)$ we denote the l-th row vector and by $R(-, k)$ the k-th column vector in the matrix representation of R.

The relation $D\text{-Cols}_n$ is the $|D|^n$-ary relation of size n, defined by

$$D\text{-Cols}_n(l, k) =_{def} D^n(k, l)$$

for all $l \in \{1, \ldots, n\}$ and $k \in \{1, \ldots, |D|^n\}$. That means the columns of $D\text{-Cols}_n$ are exactly the tuples from D^n. If D is the Boolean domain, we often write just Cols_n.

Example 3.6. The columns of the matrix representation of Cols_3 are exactly the binary numbers from 0 to $2^3 - 1 = 7$:

$$\mathrm{Cols}_3 = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

For a set $\mathcal{F}$ of D-valued functions the *$\mathcal{F}$-closure* of a relation R over D, denoted by $\mathcal{F}(R)$, is the relation $\cap_{S \in \mathrm{Inv}(\mathcal{F}), R \subseteq S} S$, i.e., the minimal superset of

R that is invariant under $\mathcal{F}$. We say R is an *$\mathcal{F}$-core* of $\mathcal{F}(R)$. For a single partial function f we write $f(R)$ instead of $\{f\}(R)$ and speak of f-closures and f-cores.

Example 3.7. We consider the Boolean relation R, defined by $R(x, y, z) = x \vee y \vee \overline{z}$. Due to [BRSV05] (see Table 3.2 as well) R generates the Boolean co-clone IV, i.e., $\mathrm{Pol}(R) = \mathrm{V}$. It follows that $\mathrm{V}(R) = R$.

We are looking for a minimal V-core of R. From Table 2.3 we know that $\mathrm{V} = [\{\vee, c_0, c_1\}]$, therefore a relation S is a V-core of R if and only if it is a $\{\vee, c_0, c_1\}$-core of R. The matrix representation of R is

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}.$$

It is easy to see that the first and the last row of R can be obtained by applaying c_0 and c_1 to some other row, for instance to the second one of R. With the notation above this means $c_0(R(2, -)) = R(1, 0)$ and $c_1(R(2, -)) = R(7, 0)$. Therefore

$$R_1 =_{def} R \setminus \{(0, 0, 0), (1, 1, 1)\}$$

is a V-core of R.

Further it holds that $R(2, -) \vee R(4, -) = R(6, -)$. So

$$R_2 =_{def} R \setminus \{(0, 0, 0), (1, 1, 1), (1, 1, 0)\}$$

is another V-core of R.

Note that all remaining rows cannot be generated from other rows in this way, therefore there is no smaller V-core for R, than R_2.

Let $s \in \mathbb{N}$. For a clone $\mathcal{C}$ we say that s is a *core-size* of the co-clone $\mathrm{I}\mathcal{C}$ if there is a relation R such that $\langle R \rangle = \mathrm{I}\mathcal{C}$ and R has a $\mathcal{C}$-core of size s. If s is a core-size of $\mathrm{I}\mathcal{C}$ and there is no $r < s$ that is a core-size of $\mathrm{I}\mathcal{C}$, then s is the *minimal core-size* of $\mathrm{I}\mathcal{C}$.

In Example 3.7 we showed that 4 is a core-size of the Boolean co-clone IV. However, we will see that 4 is not the minimal core-size of IV in the next section.

Note that not every co-clone has a core-size. If $\mathrm{I}\mathcal{C}$ has no finite base, there is no relation R such that $\langle R \rangle = \mathrm{I}\mathcal{C}$. In this case we define the minimal core-size

of $\mathrm{I}\mathcal{C}$ to be ∞. If on the contrary $\mathrm{I}\mathcal{C}$ is generated by a finite constraint language Γ, then we can easily find a relation R that generates the same co-clone as Γ: let R be the Cartesian product of all relations of Γ, then it can be proven that $\langle\Gamma\rangle = \langle R\rangle$. Therefore a co-clone has a finite core-size if and only if it has a finite base.

The following lemma is a technical result we need to identify finite weak bases. It says that if R and S are relations such that R is an $\mathcal{F}$-core of S, then every tuple from S can be generated by a single application of some function $f \in \mathcal{F}$ to the tuples of R.

Lemma 3.8. *Let R be a relation over a domain D and $\mathcal{F}$ a set of D-valued functions. Let $s =_{def} |R|$ be the size of R. Then for every $t \in \mathcal{F}(R)$ there is an s-ary function $f \in [\mathcal{F}]$ such that*

$$t = f(R(1,-),\ldots,R(s,-)).$$

Proof. We prove the lemma by induction. If $t \in R$, that means if $t = R(i,-)$ for some $1 \leq i \leq s$, then choose $f =_{def} pr_i^s$.

Otherwise there is a function $g \in \mathcal{F}$ such that $t = g(t_1,\ldots,t_k)$ for some $t_1,\ldots,t_k \in \mathcal{F}(R)$. Due to induction we know that there are s-ary functions $f_1,\ldots,f_k \in [\mathcal{F}]$ such that for every $1 \leq i \leq k$ it holds that $t_i = f_i(R(1,-),\ldots,R(s,-))$. Set

$$f =_{def} g \circ (f_1,\ldots,f_k).$$

Then we have $t = f(R(1,-),\ldots,R(s,-))$, which proves the lemma. □

Now we show that we can construct weak bases from relations of the type D-Cols_s for certain s. The idea is, that the fact that such a relation contains all tuples from D^s as columns, helps us to control partial functions.

Theorem 3.9. *Let $\mathcal{C}$ be a clone over a domain D and let $s \in \mathbb{N}$ be a core-size of $\mathrm{I}\mathcal{C}$. Then $\mathcal{C}(D\text{-}\mathrm{Cols}_s)$ is a weak base of $\mathrm{I}\mathcal{C}$.*

Proof. To show that $\mathcal{C}(D\text{-}\mathrm{Cols}_s)$ is a weak base of $\mathrm{I}\mathcal{C}$, we have to prove the following equation:

$$\langle\mathcal{C}(D\text{-}\mathrm{Cols}_s)\rangle_{\not\exists} = \mathrm{Inv}(\mathcal{I}_\cup(\mathcal{C})).$$

According to Corollary 2.10 this is equivalent to

$$\mathrm{pPol}(\mathcal{C}(D\text{-}\mathrm{Cols}_s)) = \mathcal{I}_\cup(\mathcal{C}).$$

Let $\mathcal{B} =_{def} \mathrm{pPol}(\mathcal{C}(D\text{-}\mathrm{Cols}_s))$. Note that $\mathcal{B}$ is a strong partial clone. To prove the theorem it is enough to show

$$\mathcal{B} = \mathcal{I}_\cup(\mathcal{C}).$$

First we show that $\mathcal{I}_{\cup}(\mathcal{C}) \subseteq \mathcal{B}$. Let $f \in \mathcal{I}_{\cup}(\mathcal{C})$ be an n-ary partial function. According to Theorem 3.3 we know that f is $\mathcal{C}$-total. To prove that f is a partial polymorphism of $\mathcal{C}(D\text{-Cols}_s)$ let $t_1, \ldots, t_n$ be tuples from $\mathcal{C}(D\text{-Cols}_s)$. Due to Lemma 3.8 there exist s-ary functions $h_1, \ldots, h_n \in \mathcal{C}$ such that for every $1 \leq i \leq n$ it holds

$$t_i = h_i(D\text{-Cols}_s(1,-), \ldots, D\text{-Cols}_s(s,-)).$$

Let $g =_{def} f \circ (h_1, \ldots, h_n)$. Then it follows:

$$\begin{aligned} f(t_1, \ldots, t_n) &= f \circ (h_1, \ldots, h_n)(D\text{-Cols}_s(1,-), \ldots, D\text{-Cols}_s(s,-)) \\ &= g(D\text{-Cols}_s(1,-), \ldots, D\text{-Cols}_s(s,-)). \end{aligned}$$

If g is total, then g is from $\mathcal{C}$ because f is $\mathcal{C}$-total, and it follows $f(t_1, \ldots, t_n) \in \mathcal{C}(D\text{-Cols}_s)$. If g is not total, then g is not defined on at least one of the columns of $D\text{-Cols}_s$, because every tuple from D^s is a column in $D\text{-Cols}_s$. Then $g(D\text{-Cols}_s(1,-), \ldots, D\text{-Cols}_s(s,-))$ is not defined and therefore $f(t_1, \ldots, t_n)$ is not defined either. Hence, f is a partial polymorphism of $\mathcal{C}(D\text{-Cols}_s)$.

Now we prove $\mathcal{B} \subseteq \mathcal{I}_{\cup}(\mathcal{C})$ by showing that $\mathcal{B} \in \mathcal{I}(\mathcal{C})$. That means we show that the total functions from $\mathcal{B}$ are exactly $\mathcal{C}$. Clearly it holds $\mathcal{C} \subseteq \mathcal{B}$, so we prove that every total function from $\mathcal{B}$ is a function from $\mathcal{C}$.

Let $f \in \mathcal{B}$ be a total function. Then f is a polymorphism of $\mathcal{C}(D\text{-Cols}_s)$. Since s is a core-size of $\mathrm{I}\mathcal{C}$, there exists a relation S over D such that $|S| = s$ and $\langle \mathcal{C}(S) \rangle = \mathrm{I}\mathcal{C}$. So we have $\mathrm{Pol}(\mathcal{C}(S)) = \mathcal{C}$.

We show that f is a polymorphism of $\mathcal{C}(S)$. Let n be the arity of f and $t_1, \ldots, t_n \in \mathcal{C}(S)$. According to Lemma 3.8 there exist s-ary functions $h_1, \ldots, h_n \in \mathcal{C}$ such that for every $1 \leq i \leq n$ holds

$$t_i = h_i(S(1,-), \ldots, S(s,-)).$$

Since $h_1, \ldots, h_n$ and f are polymorphisms of $\mathcal{C}(D\text{-Cols}_s)$ and the polymorphisms of a relation are closed under arbitrary composition, it follows that $g =_{def} f \circ (h_1, \ldots, h_n)$ is a polymorphism of $\mathcal{C}(D\text{-Cols}_s)$ as well. This implies that

$$t =_{def} g(D\text{-Cols}_s(1,-), \ldots, D\text{-Cols}_s(s,-)) \in \mathcal{C}(D\text{-Cols}_s).$$

We use Lemma 3.8 again: there exists an s ary function $h \in \mathcal{C}$ such that

$$t = h(D\text{-Cols}_s(1,-), \ldots, D\text{-Cols}_s(s,-)).$$

Since every element of D^s is a column of $D\text{-Cols}_s$, it holds that $g = h$. It follows that $g \in \mathcal{C}$ and therefore

$$f(t_1, \ldots, t_n) = g(S(1,-), \ldots, S(s,-)) \in \mathcal{C}(S).$$

Thus, $\mathcal{C}(S)$ is invariant under f and this implies that $f \in \mathcal{C}$. Hence, $\mathcal{B} \subseteq \mathcal{I}_{\cup}(\mathcal{C})$, which completes the proof. □

With the above theorem we can construct a weak base for every co-clone for which we know a finite base, since finite bases give us core-sizes.

Example 3.10. We construct a weak base for the Boolean co-clone IN. According to Table 2.3, it holds that $\langle \mathrm{Dup} \rangle = \mathrm{IN}$ and $[\{\neg, c_1\}] = \mathrm{N}$. Recall that $\mathrm{Dup} = \{0,1\}^3 \setminus \{(0,1,0),(1,0,1)\}$. One can verify that the relation $\{(0,0,1),(0,1,1)\}$ is an N-core of Dup, therefore 2 is a core-size of IN.

Theorem 3.9 says that $\mathrm{N}(\mathrm{Cols}_2)$ is a weak base for IN. It holds that

$$\mathrm{Cols}_2 = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}.$$

If we close this relation under $\neg$ and c_1, we get

$$\mathrm{N}(\mathrm{Cols}_2) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$

The second and the third row form Cols_2, the last row is generated by c_1, and the first, the fourth and the fifth row result from the application of $\neg$ to the last row and the rows from Cols_2.

The motivation for our interest in weak bases is that we want to use them as a tool to classify the complexity of problems in the constraint satisfaction context. However, often the more restricted closure $\langle . \rangle_{\not\equiv,\neq}$ fits more naturally to problems than the $\langle . \rangle_{\not\equiv}$-closure. The problems examined in Chapter 4 and Chapter 5 are examples for this phenomenon.

For some weak bases we can show that their properties with respect to the $\langle . \rangle_{\not\equiv}$-closure mentioned in Corollary 3.5, hold for the $\langle . \rangle_{\not\equiv,\neq}$-closure as well. To be able to give a criterion for such weak bases we introduce the notion of redundancy.

Let R be a relation over a domain D of arity n. We say R is $=$-*redundant* if there are $i, j \in \{1, \ldots, n\}$ such that $i \neq j$ and $R(-, i) = R(-, j)$. We say R is $\top$-*redundant* if there is an $i \in \{1, \ldots, n\}$ such that for all $(a_1, \ldots, a_n) \in R$ holds that for every $b \in D$ we have $(a_1, \ldots, a_{i-1}, b, a_{i+1}, \ldots, a_n) \in R$. In other words R is $=$-redundant if the matrix representation of R has two equal columns and R is $\top$-redundant if there is a relation S such that R has exactly the same columns as $S \times D$. We say R is *redundant* if R is $=$-redundant or $\top$-redundant and R is *irredundant* if R is neither $=$-redundant nor $\top$-redundant.

The following proposition shows that we do not need Eq_D-clauses to express irredundant functions.

Proposition 3.11. *Let Γ be a constraint language over a domain D and let R be an irredundant relation over D such that $R \in \langle\Gamma\rangle_{\not\equiv}$. Then it holds that $R \in \langle\Gamma\rangle_{\not\equiv,\neq}$.*

Proof. Since $R \in \langle\Gamma\rangle_{\not\equiv}$ it holds that the clause $R(x_1, \ldots, x_n)$ is equivalent to some constraint formula φ over $\Gamma \cup \{\mathrm{Eq}_D\}$ with $\mathrm{Var}(\varphi) = \{x_1, \ldots, x_n\}$. We show that we can remove all Eq_D-clauses from φ without changing its set of solutions.

Let $\mathrm{Eq}_D(x_i, x_j)$ be a clause in φ for some $i, j \in \{1, \ldots, n\}$. If $i \neq j$, then $R(-, i) = R(-, j)$, which means R is $=$-redundant. Since this contradicts the irredundancy of R it holds that $i = j$. Note that the clause then is satisfied by all assignments to x_i. If x_i does not appear in any other clause of φ, then x_i does not depend on the other variables of φ, i.e., R is $\top$-redundant in the column $R(-, i)$. Again this is a contradiction to the redundancy of R, therefore we can assume that x_i appears in some other clause in φ. But then the clause $\mathrm{Eq}_D(x_i, x_j)$ does not restrict the set of solutions for φ and we can delete it from φ without changing its set of solutions.

Thus, $\varphi \equiv \varphi'$, where φ' contains exactly the clauses from φ that are no equality clauses. Therefore $R(x_1, \ldots, x_n)$ is equivalent to the Γ-formula φ', which means $R \in \langle\Gamma\rangle_{\not\equiv,\neq}$. □

Let Γ be a weak base for a co-clone $\mathrm{I}\mathcal{C}$. If every relation from Γ is irredundant, we call Γ an *irredundant weak base* of $\mathrm{I}\mathcal{C}$.

The previous proposition yields the next corollary saying that irredundant weak bases have the same properties for the $\langle.\rangle_{\not\equiv,\neq}$-closure as weak bases have for the $\langle.\rangle_{\not\equiv}$-closure: they generate the smallest weak system without equality of all constraint languages generating the same co-clone. Note the similarity between the following corollary and point 3 in Corollary 3.5.

Corollary 3.12. *Let $\mathcal{C}$ be a clone over a domain D and let Γ be an irredundant weak base for $\mathrm{I}\mathcal{C}$. Let Γ' be a constraint language over D such that $\langle\Gamma'\rangle = \mathrm{I}\mathcal{C}$. Then it holds $\langle\Gamma\rangle_{\not\equiv,\neq} \subseteq \langle\Gamma'\rangle_{\not\equiv,\neq}$.*

Note that the weak bases that can be constructed with Theorem 3.9 are all $=$-irredundant, because D-Cols_s has no double columns. However, these weak bases are not necessarily $\top$-irredundant.

3.2 Boolean Weak Bases

Since for a core-size s the arity of D-Cols_s is $|D|^s$, our weak bases can become very large quickly. To be able to work with weak bases it is therefore in our

interest to find minimal core-sizes of co-clones. In this section we give the minimal core-sizes for all Boolean co-clones.

The results from this section can be seen in Table 3.2. It shows for each Boolean co-clone its minimal core-size. It also lists bases discovered by Böhler et al. [BRSV05] for the co-clones. Note that most of these bases are not weak bases.

Theorem 3.13. *The minimal core-sizes given in Table 3.2 for the Boolean co-clones are correct.*

Proof. Due to [BRSV05] the eight co-clones IS_0, IS_{00}, IS_{01}, IS_{02}, IS_1, IS_{10}, IS_{11}, and IS_{12} do not have finite bases, therefore their minimal core-size is ∞.

As an example for the other cases we prove that the minimal core-size of IV is 2.

We start with showing that 2 is a core-size of the co-clone IV. Let $R =_{def} \mathrm{Cols}_2 = \{(0,0,1,1),(0,1,0,1)\}$. We show that $\mathrm{V}(R)$ generates IV. The matrix representation of $\mathrm{V}(R)$ is

$$\mathrm{V}(R) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$

Since $\mathrm{V} \subseteq \mathrm{Pol}(\mathrm{V}(R))$, it holds $\langle \mathrm{V}(R) \rangle \subseteq \mathrm{IV}$ according to Proposition 2.8. One can verify that the following equivalence is true:

$$\exists t,u,v,w R(t,x,y,v) \wedge R(u,w,z,v) \equiv x \vee y \vee \overline{z}.$$

Since the relation S defined by

$$S(x,y,z) \equiv x \vee y \vee \overline{z}$$

forms a base of IV (see Table 3.2) it holds that $\langle \mathrm{V}(R) \rangle = \mathrm{IV}$. Thus, IV has 2 as a core-size.

Now assume that 1 is a core-size of IV. Then $\mathrm{V}(\mathrm{Cols}_1)$ is a weak base of IV due to Theorem 3.9. In particular we have $\langle \mathrm{V}(\mathrm{Cols}_1) \rangle = \mathrm{IV}$. But it holds $\mathrm{V}(\mathrm{Cols}_1) = \{(0,0),(0,1),(1,1)\} = \mathrm{Imp}$ and $\langle \mathrm{Imp} \rangle = \mathrm{IM}$ due to Table 3.2. Hence, 1 is not a core-size of IV, so 2 is the minimal core-size of IV.

The minimal core-sizes for the other Boolean co-clones can be verified in a similar way: to prove that the stated minimal core-size s for a co-clone $\mathrm{I}\mathcal{C}$ really is a core-size one shows that $\mathcal{C}(\mathrm{Cols}_s)$ is a co-clone base of $\mathrm{I}\mathcal{C}$. For the minimality of s one shows that $\mathcal{C}(\mathrm{Cols}_{s-1})$ is not a co-clone base of $\mathrm{I}\mathcal{C}$, by

Co-Clone	**Min. core-size**	**Base**
IBF	1	$\{\mathrm{Eq}\}$
IR_0	1	$\{\mathrm{C}_0\}$
IR_1	1	$\{\mathrm{C}_1\}$
IR_2	1	$\{\mathrm{C}_0, \mathrm{C}_1\}$
IM	1	$\{\mathrm{Imp}\}$
IM_0	2	$\{\mathrm{Imp}, \mathrm{C}_0\}$
IM_1	2	$\{\mathrm{Imp}, \mathrm{C}_1\}$
IM_2	3	$\{\mathrm{Imp}, \mathrm{C}_0, \mathrm{C}_1\}$
$\mathrm{IS}_0^m,\ m \geq 2$	m	$\{\mathrm{Or}^m\}$
IS_0	∞	$\{\mathrm{Or}^m \mid m \geq 1\}$
$\mathrm{IS}_{02}^m,\ m \geq 2$	m	$\{\mathrm{Or}^m, \mathrm{C}_0\}$
IS_{02}	∞	$\{\mathrm{Or}^m \mid m \geq 1\} \cup \{\mathrm{C}_0\}$
$\mathrm{IS}_{01}^m,\ m \geq 2$	m	$\{\mathrm{Or}^m, \mathrm{Imp}\}$
IS_{01}	∞	$\{\mathrm{Or}^m \mid m \geq 1\} \cup \{\mathrm{Imp}\}$
IS_{00}^2	3	$\{\mathrm{Or}^2, \mathrm{Imp}, \mathrm{C}_0\}$
$\mathrm{IS}_{00}^m,\ m \geq 3$	m	$\{\mathrm{Or}^m, \mathrm{Imp}, \mathrm{C}_0\}$
IS_{00}	∞	$\{\mathrm{Or}^m \mid m \geq 1\} \cup \{\mathrm{Imp}, \mathrm{C}_0\}$
$\mathrm{IS}_1^m,\ m \geq 2$	m	$\{\mathrm{dual}(\mathrm{Or}^m)\}$
IS_1	∞	$\{\mathrm{dual}(\mathrm{Or}^m) \mid m \geq 1\}$
$\mathrm{IS}_{12}^m,\ m \geq 2$	m	$\{\mathrm{dual}(\mathrm{Or}^m), \mathrm{C}_1\}$
IS_{12}	∞	$\{\mathrm{dual}(\mathrm{Or}^m) \mid m \geq 1\} \cup \{\mathrm{C}_1\}$
$\mathrm{IS}_{11}^m,\ m \geq 2$	m	$\{\mathrm{dual}(\mathrm{Or}^m), \mathrm{Imp}\}$
IS_{11}	∞	$\{\mathrm{dual}(\mathrm{Or}^m) \mid m \geq 1\} \cup \{\mathrm{Imp}\}$
IS_{10}^2	3	$\{\mathrm{dual}(\mathrm{Or}^2), \mathrm{Imp}, \mathrm{C}_1\}$
$\mathrm{IS}_{10}^m,\ m \geq 3$	m	$\{\mathrm{dual}(\mathrm{Or}^m), \mathrm{Imp}, \mathrm{C}_1\}$
IS_{10}	∞	$\{\mathrm{dual}(\mathrm{Or}^m) \mid m \geq 1\} \cup \{\mathrm{Imp}, \mathrm{C}_1\}$
ID	1	$\{\mathrm{Odd}^2\}$
ID_1	2	$\{\mathrm{Odd}^2, \mathrm{C}_1\}$
ID_2	3	$\{\mathrm{Odd}^2, \mathrm{Imp}\}$
IL	2	$\{\mathrm{Even}^4\}$
IL_0	2	$\{\mathrm{Even}^4, \mathrm{C}_0\}, \{\mathrm{Even}^3\}$
IL_1	2	$\{\mathrm{Even}^4, \mathrm{C}_1\}, \{\mathrm{Odd}^3\}$
IL_2	3	$\{\mathrm{Even}^4, \mathrm{C}_0, \mathrm{C}_1\}, \{\mathrm{Even}^3, \mathrm{C}_1\}$
IL_3	3	$\{\mathrm{Even}^4, \mathrm{Odd}^2\}, \{\mathrm{Odd}^4\}$
IV	2	$\{x \vee y \vee \overline{z}\}$
IV_0	2	$\{x \vee y \vee \overline{z}, \overline{x}\}$
IV_1	3	$\{x \vee y \vee \overline{z}, x\}$
IV_2	3	$\{x \vee y \vee \overline{z}, x, \overline{x}\}$
IE	2	$\{\overline{x} \vee \overline{y} \vee z\}$
IE_0	3	$\{\overline{x} \vee \overline{y} \vee z, \overline{x}\}$
IE_1	2	$\{\overline{x} \vee \overline{y} \vee z, x\}$
IE_2	3	$\{\overline{x} \vee \overline{y} \vee z, x, \overline{x}\}$
IN	2	$\{\mathrm{Dup}\}$
IN_2	3	$\{\mathrm{Dup}, \mathrm{Even}^4, \mathrm{Odd}^2\}, \{\mathrm{Nae}\}$
II	2	$\{\mathrm{Even}^4, \mathrm{Imp}\}$
II_0	2	$\{\mathrm{Even}^4, \mathrm{Imp}, \mathrm{C}_0\}$
II_1	2	$\{\mathrm{Even}^4, \mathrm{Imp}, \mathrm{C}_1\}$
II_2	3	$\{\mathrm{Even}^4, \mathrm{Imp}, \mathrm{C}_0, \mathrm{C}_1\}$, {1-in-3}

Table 3.1: Minimal core-sizes and bases for all Boolean co-clones

proving that $\mathcal{C}(\mathrm{Cols}_{s-1})$ generates a co-clone smaller than $\mathrm{I}\mathcal{C}$. Since it is easy to show that every co-clone which has $r \in \mathbb{N}$ as a core-size, has $r+1$ as a core-size as well, this is sufficient to prove that s is the minimal core-size of $\mathrm{I}\mathcal{C}$. □

The concept of weak bases we have developed is essential for the classifications in the next two chapters. It allows, even if the co-clone closure cannot be shown to preserve the complexity of the particular problem, to use the co-clone structure to obtain hardness results, by switching over from a co-clone $\mathrm{I}\mathcal{C}$ to the smallest corresponding weak system $\mathrm{Inv}(\mathcal{I}_\cup(\mathcal{C}))$. But we have to be careful in one point: if we have two co-clones $\mathrm{I}\mathcal{C}$ and $\mathrm{I}\mathcal{D}$ such that $\mathrm{I}\mathcal{C} \subseteq \mathrm{I}\mathcal{D}$, then it does not necessarily hold that $\mathrm{Inv}(\mathcal{I}_\cup(\mathcal{C})) \subseteq \mathrm{Inv}(\mathcal{I}_\cup(\mathcal{D}))$. For example we have $\mathrm{IR}_1 \subseteq \mathrm{IR}_2$, but the weak base $\{(0,1),(1,1)\}$ of IR_1 is not in the weak system generated by $\{(0,1)\}$ which is a weak base for IR_2. That this fact effects the use of weak bases for classifications can be seen in Chapter 4.

It is an interesting task for future research to find out which pairs of co-clones hand down their inclusion relation to their smallest weak systems. Another issue that arises is the inclusion structure of the weak systems corresponding to the same co-clone.

Chapter 4

Balanced Satisfiability

In this chapter we apply the tools from Chapter 3 to classify the complexity of the balanced satisfiability problem which is an example for a *global satisfiability problem*. In a global satisfiability problem we have additionally to local constraints expressed by constraint clauses a global condition restricting the solutions of a formula. In the case of balanced satisfiability we require the solutions to set exactly on half of the variables to 1.

Let Γ be a constraint language over the Boolean domain $\{0, 1\}$ and let φ be a constraint formula over Γ. A *balanced assignment* for φ is an assignment $I : \mathrm{Var}(\varphi) \to D$ that assigns 0 to the same number of variables as 1, that means it fullfills $|\{x \in \mathrm{Var}(\varphi) \mid I(x) = 0\}| = |\{x \in \mathrm{Var}(\varphi) \mid I(x) = 1\}|$. If I additionally satisfies φ we call I a *balanced solution* of φ.

The balanced satisfiability problem for Γ-formulas is defined as follows:

Problem:	BAL-CSP(Γ)
Input:	a Γ-formula φ
Question:	does φ have a balanced solution?

Additionally we look at the counting version of this problem, i.e., the question how many balanced solutions a given Γ-formula has.

Problem:	#BAL-CSP(Γ)
Input:	a Γ-formula φ
Question:	how many balanced solutions does φ have?

Partial results for the decision problem and an optimization version of balanced satisfiability have been achieved by Bazgan and Karpinski [BK05]. The aim of this chapter is to prove a full complexity classification for both problems, saying that for every finite Boolean constraint language BAL-CSP(Γ) is in P or NP-complete and #BAL-CSP(Γ) is in FP or complete for #P.

We start with stating canonical upper complexity bounds and observing that the complexity of both problems we look at is invariant under the $\langle . \rangle_{\sharp,\neq}$-closure. We then show the polynomial time results in Section 4.1. Our focus lies on the hardness results in the later sections: in Section 4.2 we show that both problems are NP-hard or #P-hard for some special one-element constraint languages, which we need in many later proofs. Then we deal with general finite constraint languages in the next two sections. All those hardness results follow the same scheme: to show that for all constraint languages Γ generating a fixed Boolean co-clone $\mathrm{I}\mathcal{C}$ the problems BAL-CSP(Γ) and #BAL-CSP(Γ) are hard for NP or #P, we prove the hardness for an irredundant weak base of $\mathrm{I}\mathcal{C}$. For the proofs in Section 4.3 we do not need to construct an irredundant weak bases explicitly and we can handle several co-clones at once, because their irredundant weak bases share some properties. In Section 4.4 we work with concrete irredundant weak bases and cover only one co-clone per proof. Allthough the arguments are very similar their is no obvious way to unify these proofs. We give the appearing constructions all in detail as an example how to work with irredundant weak bases. These non-unifiable cases include among others the non-Schaefer co-clones. The reason for their uncomfortable behavior could lie in a fact mentioned in Chapter 3: inclusions of co-clones do not necessarily transfer to the weaker closure operators.

4.1 Basic Facts and Easy Cases

First we state the obvious upper complexity bounds for BAL-CSP(Γ) and #BAL-CSP(Γ).

Proposition 4.1. *For every finite constraint language* Γ *over* $\{0,1\}$ *it holds that* BAL-CSP(Γ) $\in$ NP *and* #BAL-CSP(Γ) $\in$ #P.

Proof. Given a Γ-formula φ, an algorithm can easily verify if a guessed assignment for φ is a balanced solution for φ. Therefore we can construct a non-deterministic polynomial time Turing Machine φ that branches out for every possible Boolean assignment to the variables of the input formula and accepts in a branch if the according assignment is a balanced solution for φ. This proves BAL-CSP(Γ) $\in$ NP and #BAL-CSP(Γ) $\in$ #P. □

To make sure that we can use the tools developed in Chapter 3, we prove that the computational complexity of BAL-CSP(Γ) depends only on the weak system without equality generated by Γ. The result follows directly from the fact that two equivalent formulas have the same solutions.

Proposition 4.2. *Let* Γ_1 *and* Γ_2 *be finite constraint languages over* $\{0,1\}$ *such that* $\Gamma_1 \subseteq \langle\Gamma_2\rangle_{\not\exists,\neq}$. *Then it holds* #BAL-CSP(Γ_1) $\leq_!^{log}$ #BAL-CSP(Γ_2) *and* BAL-CSP(Γ_1) $\leq_m^{log}$ BAL-CSP(Γ_2).

Proof. Let φ be a constraint formula over Γ_1. According to Proposition 2.5 we can compute a Γ_2-formula φ', that is equivalent to φ, using only logarithmic space. Since φ and φ' have the same set of solutions, they have the same set of balanced solutions as well, therefore it holds #BAL-CSP(Γ_1) $\leq_!^{\log}$ #BAL-CSP(Γ_2) and in particular BAL-CSP(Γ_1) $\leq_m^{\log}$ BAL-CSP(Γ_2). □

Note that the above proof cannot be generalized canonically to work with the weakend prerequisite $\Gamma_1 \subseteq \langle\Gamma_2\rangle_{\not\exists}$ or even $\Gamma_1 \subseteq \langle\Gamma_2\rangle$ instead of $\Gamma_1 \subseteq \langle\Gamma_2\rangle_{\not\exists,\neq}$. Therefore, at this point, we cannot apply the classical methods involving only the co-clone closure. However, in the end of this chapter, we will see that the proposition stays true with this modified prerequisites, i.e, that the complexity of balanced satisfiability depends only on the co-clone generated by the according constraint language.

A helpful property of balanced satisfiability is that we can exploit the symmetry in Post's Lattice: the next proposition says that the complexity of our problem is invariant under dualization.

Proposition 4.3. *Let* Γ *be a finite constraint language over* $\{0,1\}$*, then it holds* #BAL-CSP(dual(Γ)) $\leq_!^{log}$ #BAL-CSP(Γ)

Proof. Let φ be a Γ-formula and let φ' be the dual(Γ)-formula obtained by replacing every clause $R(x_1,\ldots,x_n)$ from φ by dual(R)$(x_1,\ldots,x_n)$. It is easy to see that an assignment $I : \mathrm{Var}(\varphi) \to \{0,1\}$ satisfies φ if and only if the assignment $I' : \mathrm{Var}(\varphi) \to \{0,1\}$, defined by $I'(x) = \neg I(x)$ for all $x \in \mathrm{Var}(\varphi)$, satisfies φ'. Since I is balanced if and only if I' is, and since the replacement of the clauses is a local operation that works in logarithmic space, this proves the reduction stated in the proposition. □

We now show that BAL-CSP(Γ) and #BAL-CSP(Γ) are solvable in polynomial time for finite Boolean constraint languages Γ, such that $\langle\Gamma\rangle \subseteq \mathrm{ID}_1$. As an intermediate problem we use a version of SUBSET-SUM, which is the question whether, given a sequence of natural numbers $k_1,\ldots,k_n$ and a natural number S, there is a subsequence $k_{i_1},\ldots,k_{i_m}$ of $k_1,\ldots,k_n$, such that the sum over its elements is exactly S. This problem is well known to be NP-complete [Kar72]. However, if the numbers $k_1,\ldots,k_n$ in the input are given in a unary representation, then we can solve the problem in polynomial time [GJ79]. We give the formal definition for the counting version of this variant of SUBSET-SUM:

Problem: #UNARY-SUBSET-SUM
Input: a sequence $1^{k_1}, \ldots, 1^{k_n}$ for some $n, k_1, \ldots, k_n \in \mathbb{N}$ and a natural number S.
Question: how many sets $I \subseteq \{1, \ldots, n\}$ exist, such that $\sum_{i \in I} k_i = S$?

By 1^k we denote the word of length k over the alphabet $\{1\}$, i.e., 1^0 is the empty word and $1^1 = 1$, $1^2 = 11$, $1^3 = 111$, etc.

We show that we can solve #UNARY-SUBSET-SUM in polynomial time in the next theorem. The algorithm in the proof is strongly based on a pseudo-polynomial time algorithm for PARTITION from [GJ79].

Lemma 4.4. #UNARY-SUBSET-SUM $\in$ FP.

Proof. Let $1^{k_1}, \ldots, 1^{k_n}$ for some $n, k_1, \ldots, k_n \in \mathbb{N}$ and $S \in \mathbb{N}$ be an instance for #UNARY-SUBSET-SUM. The following algorithm computes for every $0 \leq i \leq n$ and $0 \leq j \leq S$ the number of sets $I \subseteq \{1, \ldots, i\}$ such that $\sum_{i \in I} k_i = j$ and stores this information in a matrix T. The algorithm fills the matrix line by line and relies on earlyer entries when computing the current one. The information we will looking for will be in $T(n, S)$. This approach is known as dynamic programming.

```
1: let T be an (n + 1) × (S + 1)-matrix
2: T(0, 0) := 1
3: for j = 1, ..., S do
4:    T(0, j) := 0
5: end for
6: for i = 1, ..., n do
7:    for j = 0, ..., S do
8:       if j < k_i then
9:          T(i, j) := T(i − 1, j)
10:      else
11:         T(i, j) := T(i − 1, j) + T(i − 1, j − k_i)
12:      end if
13:   end for
14: end for
15: return T(n, S)
```

The algorithm fills a $(n + 1) \times (S + 1)$-matrix and needs only polynomial time for each entry, so the algorithm is polynomial in the length of the instance which is $O(k_1 + \cdots + k_n + S)$. Note that without using unary coding the length of the input is $O(\log(k_1 \cdots k_n S))$ and the running time of the above algorithm is exponential in the size of the input. □

The following result covers all polynomial time cases for our problems. In the next section we will see that for all other cases we prove NP-hardness and #P-hardness respectively.

Theorem 4.5. *Let* Γ *be a finite constraint language over* $\{0,1\}$ *such that* $\langle\Gamma\rangle \subseteq \mathrm{ID}_1$. *Then* BAL-CSP($\Gamma$) $\in$ P *and* #BAL-CSP(Γ) $\in$ FP

Proof. Due to [CKZ05] it holds that the constraint language

$$\Sigma =_{def} \left\{\mathrm{C}_0, \mathrm{C}_1, \mathrm{Even}^2, \mathrm{Odd}^2\right\}$$

is a so called *plain base* for ID_1, that means it has the following properties: $\langle\Sigma\rangle = \mathrm{ID}_1$ and $\langle\Gamma\rangle_{\nexists,\neq} \subseteq \langle\Sigma\rangle_{\nexists,\neq}$. Therefore it follows from Proposition 4.2 that #BAL-CSP(Γ) $\leq_!^{\log}$ #BAL-CSP(Σ) and BAL-CSP(Γ) $\leq_m^{\log}$ BAL-CSP(Σ) is true. So, it is sufficient to show #BAL-CSP(Σ) $\in$ FP to prove the theorem.

Let φ be a constraint-formula over Σ. Since Σ is Schaefer it follows from Schaefer's Theorem (see Theorem 2.11) that we can check in polynomial time whether φ is satisfiable. If this is the case, we can partition $\mathrm{Var}(\varphi)$ in classes $X_0, \ldots, X_n, Y_0, \ldots, Y_n$ for some $n \in \mathbb{N}$, such that the following is true for every $v \in \mathrm{Var}(\varphi)$:

- if $\mathrm{C}_1(v)$ is a clause in φ, then $v \in X_0$;
- if $\mathrm{C}_0(v)$ is a clause in φ, then $v \in Y_0$;
- if φ has neither a C_0-clause nor a C_1-clause, then $X_0 = Y_0 = \emptyset$;
- if $v \in X_i$ (resp. $v \in Y_i$) for some $0 \leq i \leq n$, then X_i (resp. Y_i) is the set of all variables v' of φ such that for every solution I for φ holds $I(v) = I(v')$;
- if $v \in X_i$ (resp. $v \in Y_i$) for some $0 \leq i \leq n$, then Y_i (resp. X_i) is the set of all variables v' of φ such that for every solution I for φ holds $I(v) \neq I(v')$.

Note that we can construct this partition in polynomial time, because for two variables v and v' from $\mathrm{Var}(\varphi)$ it holds that $I(v) = I(v')$ for each solution I of φ, if and only if v and v' are connected by a path of clauses that includes an even number of Odd^2 clauses. And it holds that $I(v) \neq I(v')$ for each solution I of φ, if and only if v and v' are connected by a path of clauses that includes an odd number of Odd^2 clauses. By a path of clauses we mean a set of clauses $\{c_1, \ldots, c_m\}$, such that for every $1 \leq i \leq m-1$ it holds that $\mathrm{Var}(c_i) \cap \mathrm{Var}(c_{i+1}) \neq \emptyset$, or that c_i and c_{i+1} are both C_a-clauses for an $a \in \{0,1\}$.

Assuming that we do not have a pair of empty classes (X_i, Y_i) for $1 \leq i \leq n$, it holds that the number of solutions for φ equals 2^n, because for every pair (X_i, Y_i) with $1 \leq i \leq n$ we can chose independently whether we set the variables from X_i to 1 and from Y_i to 0 or the other way round. Note that the values for the variables in X_0 and Y_0 are fixed.

Let $k_i =_{def} |X_i| - |Y_i|$ for every $i \in \{1, \ldots, n\}$. Without loss of generality assume that $k_i \geq 0$ for every $i \in \{1, \ldots, n\}$, otherwise exchange X_i with Y_i. Let I be a solution for φ and let I^X be the set of all indices $i \neq 0$ such that I maps the variables from X_i to 1, i.e.,

$$I^X =_{def} \{i \mid 1 \leq i \leq n \text{ and } I(x) = 1 \text{ for all } x \in X_i\}$$

Let $K(I)$ be the number of variables $v \in \mathrm{Var}(\varphi)$ such that $I(v) = 1$. Then it holds:

$$K(I) = |X_0| + \sum_{i=1}^{n} |Y_i| + \sum_{i \in I^X} k_i$$

It follows that I is balanced if and only if

$$\frac{1}{2}|\operatorname{Var}(\varphi)| - \sum_{i=1}^{n} |Y_i| - |X_0| = \sum_{i \in I^X} k_i.$$

Thus the number of balanced solutions for φ is exactly the number of subsets $I^X \subseteq \{1, \ldots, n\}$ such that I^X satisfies the previous equation. Note that $\sum_{1=1}^{n} k_i \leq |\operatorname{Var}(\varphi)|$, therefore the length of the string 1^{k_i} is polynomial in the length of φ. Hence, since #UNARY-SUBSET-SUM is in FP due to Lemma 4.4, we can compute the number of balanced solutions for φ in polynomial time.

Note that for some $K \in \mathbb{N}$ we can compute the number of all solutions I for φ, that map exactly K variables of φ to 1, in polynomial time as well. We just have to replace $\frac{1}{2}|\operatorname{Var}(\varphi)|$ in the last equation by K.

□

4.2 Hardness Results for Basic Relations

In this section we look at the three relations Imp, Or^2 and Odd^3 to be able to use them later to obtain complexity results for more general constraint languages.

Lemma 4.6. BAL-CSP(Imp) *is* NP*-hard and* #BAL-CSP(Imp) *is* #P*-hard under counting reductions.*

Proof. First we prove the NP-hardness of BAL-CSP(Imp). For that we make a reduction from the following problem:

Problem:	K-CLOSURE
Input:	a directed graph $G = (V, E)$ and $k \in \mathbb{N}$
Question:	Is there a $V' \in V$ such that $\|V'\| = k$ and for all $(u, v) \in E$ it holds $u \in V'$ or $v \notin V'$?

Speaking more intuitively the question is whether there is a k-element subset V' of V such that no edge goes from $V \setminus V'$ to V'. Due to [GJ79] K-CLOSURE is NP-complete.

We show K-CLOSURE $\leq_m^{\log}$ BAL-CSP(Imp). Let $G = (V, E)$ be a directed graph and $k \in \mathbb{N}$. Let $n =_{def} |V|$. We construct a constraint formula over $\{\text{Imp}\}$ with variables $X =_{def} V \cup \{t_1, \ldots, t_k, f_1, \ldots, f_{n-k}\}$, where $t_1, \ldots, t_k, f_1, \ldots, f_{n-k}$ are all distinct variables and not from V. We set

$$\varphi =_{def} \bigwedge_{(u,v)\in E} \text{Imp}(u, v) \wedge \bigwedge_{i=1}^{k} \bigwedge_{x\in X} \text{Imp}(x, t_i) \wedge \bigwedge_{i=1}^{n-k} \bigwedge_{x\in X} \text{Imp}(f_i, x).$$

Let $V' \subseteq V$ such that $|V'| = k$ and for every $(u, v) \in E$ it holds that $u \in V'$ or $v \notin V'$. It is easy to see that the assignment $I : X \to \{0, 1\}$ defined by

$$I(x) =_{def} \begin{cases} 0 & \text{if } x \in V' \cup \{f_1, \ldots, f_{n-k}\} \\ 1 & \text{otherwise} \end{cases}$$

is a balanced solution for φ.

Now let $I : X \to \{0, 1\}$ be a balanced solution for φ. Assume $I(t_i) = 0$ for an $i \in \{1, \ldots, k\}$. Then, because of the clauses $\bigwedge_{x\in X} \text{Imp}(x, t_i)$, it follows that $I(x) = 0$ for every $x \in X$. This contradicts the fact that I is balanced, therefore it holds $I(t_i) = 1$ for every $i \in \{1, \ldots, k\}$ and analogously it follows $I(f_i) = 0$ for every $i \in \{1, \ldots, n - k\}$.

Because I is balanced I maps k variables from V to 0 and $n - k$ variables form V to 1. Let

$$V' =_{def} \{x \in V \mid I(x) = 0\}.$$

Then we have $|V'| = k$. Let $(u, v) \in E$ and assume $u \notin V'$ and $v \in V'$. That means $I(u) = 1$ and $I(v) = 0$. Since $\text{Imp}(u, v)$ is a clause from φ we have a contradiction. Hence, V' satisfies all properties for a k-closure.

Since φ can be constructed in logarithmic space, it follows

$$\text{K-CLOSURE} \leq_m^{\log} \text{BAL-CSP(Imp)}.$$

Thus, BAL-CSP(Imp) is NP-hard.

Now we show that #BAL-CSP(Imp) is #P-hard by proving

$$\#\text{CSP(Imp)} \leq^{\log}_{!} \#\text{BAL-CSP(Imp)}.$$

Let

$$\varphi =_{def} \text{Imp}(x_1, y_1) \wedge \cdots \wedge \text{Imp}(x_n, y_n).$$

Let $k =_{def} |\operatorname{Var} \varphi|$ and $z_1, \ldots, z_k$ be new and distinct variables. We set:

$$\varphi' =_{def} \varphi \wedge \bigwedge_{i=1}^{k-1} \text{Imp}(z_i, z_{i+1}).$$

Obviously every balanced solution for φ' can be restricted to a solution for φ.

For the converse direction we show that every solution for φ can be extended in exactly one way to a balanced solution for φ'. Let $I : \text{Var}(\varphi) \to \{0, 1\}$ be a solution for φ. Let $k_0 =_{def} |\{x \in \text{Var}(\varphi) \mid I(x) = 0\}|$ and $k_1 =_{def} |\{x \in \text{Var}(\varphi) \mid I(x) = 1\}|$. If we want to extend I to a balanced solution for φ', then we have to assign 0 to k_1 variables from $\{z_1, \ldots, z_k\}$ and 1 to k_0 variables from $\{z_1, \ldots, z_k\}$. In order to satisfy all clauses of the form $\text{Imp}(z_i, z_{i+1})$ we have to set $I(z_i) = 0$ if $i \leq k_1$ and $I(z_i) = 1$ otherwise.

Thus, there is a one-to-one correspondence between solutions for φ and balanced solutions for φ'. Since φ' can be constructed from φ in logarithmic space, this proves the reduction $\#\text{CSP(Imp)} \leq^{\log}_{!} \#\text{BAL-CSP(Imp)}$. Due to Theorem 2.12 it holds that #CSP(Imp) is #P-hard, therefore it follows #P-hardness of #BAL-CSP(Imp). □

The NP-hardness for BAL-CSP(Or^2) was proven in [BK05], we add the #P-hardness for the counting problem.

Lemma 4.7. BAL-CSP(Or^2) *is* NP*-hard and* #BAL-CSP(Or^2) *is* #P*-hard under counting reductions.*

Proof. Bazgan and Karpinski showed in [BK05] that BAL-CSP(dual(Or^2)) is NP-hard, therefore due to Proposition 4.3 it also holds that BAL-CSP(Or^2) is NP-hard.

For the #P-hardness of #BAL-CSP(Or^2) we show the following reduction:

$$\#\text{CSP}(\text{Or}^2) \leq^{\log}_{!} \#\text{BAL-CSP}(\text{Or}^2).$$

Let φ be an Or^2-formula. For every $x \in \text{Var}(\varphi)$ let x' be a new variable. We define

$$\varphi' =_{def} \varphi \wedge \bigwedge_{x \in \text{Var}(\varphi)} \text{Or}^2(x, x').$$

Note that φ' can be constructed from φ in logarithmic space.

It is easy to see that a balanced assignment $I : \mathrm{Var}(\varphi') \to \{0,1\}$ satisfies the subformula $\bigwedge_{x\in\mathrm{Var}(\varphi)} \mathrm{Or}^2(x,x')$ if and only if for every $x \in \mathrm{Var}(\varphi)$ holds $I(x) \neq I(x')$. Therefore every solution $I : \mathrm{Var}(\varphi) \to \{0,1\}$ for φ can be extended to a balanced solution for φ' only in one way: by setting $I(x') \neq_{def} I(x)$ for every $x \in \mathrm{Var}(\varphi)$. Because obviously every balanced solution for φ' satisfies φ, we have a one-to-one correspondence between solutions for φ and balanced solutions for φ'. Thus, $\#\mathrm{CSP}(\mathrm{Or}^2) \leq^{\log}_{!} \#\text{BAL-CSP}(\mathrm{Or}^2)$.

Since $\langle \mathrm{Or}^2 \rangle = \mathrm{IS}^2_0$ (see Table 3.2), it holds that $\#\mathrm{CSP}(\mathrm{Or}^2)$ is #P-hard due to Theorem 2.12. This gives us #P-hardness for $\#\text{BAL-CSP}(\mathrm{Or}^2)$. □

Again, the NP-hardness for $\text{BAL-CSP}(\mathrm{Odd}^3)$ was proven in [BK05]. Note that in the proof for the #P-hardness of $\#\text{BAL-CSP}(\mathrm{Odd}^3)$ we reduce a non-Schaefer to a Schaefer case.

Lemma 4.8. $\text{BAL-CSP}(\mathrm{Odd}^3)$ *is* NP*-hard and* $\#\text{BAL-CSP}(\mathrm{Odd}^3)$ *is* #P*-hard under counting reductions.*

Proof. The NP-hardness of $\text{BAL-CSP}(\mathrm{Odd}^3)$ was already proven in [BK05]. For the #P-hardness of the counting problem it is sufficient to show that $\#\text{1-in-3} \leq^{\log}_{!} \#\text{BAL-CSP}(\mathrm{Odd}^3)$, because #1-in-3 is hard for #P due to Table 3.2 and the result from Creignou and Hermann stated in Theorem 2.12. Note that, since CSP(1-in-3) is an NP-complete problem due to Theorem 2.11, the following reduction is also an alternative proof for the NP-hardness of $\text{BAL-CSP}(\mathrm{Odd}^3)$.

Let

$$\varphi =_{def} \bigwedge_{i=1}^{n} \text{1-in-3}(x_i, y_i, z_i)$$

be a constraint formula over {1-in-3}. We construct an Odd^3-formula using additionally to the variables appearing in φ the following new and distinct variables: a_i, b_i, c_i, d_i for every $1 \leq i \leq n$; t^i, f^i for every $1 \leq i \leq k$ where $k =_{def} 2|\,\mathrm{Var}(\varphi)| + 4n$; and v' for every $v \in \varphi$.

We set:

$$\begin{aligned}\varphi' =_{def} \bigwedge_{i=1}^{n} \big\{ &\mathrm{Odd}^3(x_i, y_i, z_i) \wedge \mathrm{Odd}^3(d_i, d_i, d_i) \wedge \\ &\mathrm{Odd}^3(d_i, x_i, a_i) \wedge \mathrm{Odd}^3(d_i, y_i, b_i) \wedge \mathrm{Odd}^3(d_i, z_i, c_i) \,\big\} \\ \wedge \bigwedge_{i=1}^{k} \; &\mathrm{Odd}^3(t^i, t^i, t^i) \wedge \mathrm{Odd}^3(t^i, f^i, f^1) \\ \wedge \bigwedge_{v\in\mathrm{Var}(\varphi)} \; &\mathrm{Odd}^3(f^1, v, v').\end{aligned}$$

Note that $|\operatorname{Var}(\varphi')| = 2|\operatorname{Var}(\varphi)| + 4n + 2k = 3k$. Let $I : \operatorname{Var}(\varphi') \to \{0,1\}$ be a balanced solution for φ'. We show that I is uniquely determined by its values for $\operatorname{Var}(\varphi)$. For every $1 \leq i \leq k$, it holds that $I(t^i) = 1$, because $\operatorname{Odd}^3(t^i, t^i, t^i)$ is a clause from φ', and $I(f^i) = I(f^1)$, because $\operatorname{Odd}^3(t^i, f^i, f^1)$ is a clause from φ'. Now assume $I(f^1) = 1$. Then it follows $I(f^1) = \cdots = I(f^k) = I(t^1) = \cdots = I(t^k) = 1$. But since $2k > \frac{1}{2}|\operatorname{Var}(\varphi)| = \frac{3}{2}k$ this contradicts the prerequisite that I is balanced. Therefore we have

$$I(f^1) = \cdots = I(f^k) = 0.$$

With this the clauses $\operatorname{Odd}^3(f^1, v, v')$ give us $I(v) \neq I(v')$ for every $v \in \operatorname{Var}(\varphi)$. So I is already balanced on $\operatorname{Var}(\varphi) \cup \{v' \mid v \in \operatorname{Var}(\varphi)\}$ and as well on the set $\{t^1, \ldots, t^k, f^1, \ldots, f^k\}$. It follows that I is also balanced on the rest of the variables of φ', i.e., on $\{a_1, b_1, c_1, d_1, \ldots, a_n, b_n, c_n, d_n\}$.

Since we have the clause $\operatorname{Odd}^3(d_i, d_i, d_i)$ for every $1 \leq i \leq n$, it holds that

$$I(d_1) = \cdots = I(d_n) = 1.$$

Therefore the clauses $\operatorname{Odd}^3(d_i, x_i, a_i)$, $\operatorname{Odd}^3(d_i, y_i, b_i)$, and $\operatorname{Odd}^3(d_i, z_i, c_i)$, give us $I(x_i) = I(a_i)$, $I(y_i) = I(b_i)$, and $I(z_i) = I(c_i)$. Thus, for every variable from φ' its value under I is uniquely determined by $I|_{\operatorname{Var}(\varphi)}$.

We now show that $I|_{\operatorname{Var}(\varphi)}$ is a solution for φ. Assume there is a clause 1-in-3(x_i, y_i, z_i) in φ that is not satisfied by $I|_{\operatorname{Var}(\varphi)}$. Since $\operatorname{Odd}^3(x_i, y_i, z_i)$ is a clause in φ' and $\operatorname{Odd}^3 = \text{1-in-3} \cup \{(1,1,1)\}$, it holds that $I(x_i) = I(y_i) = I(z_i) = 1$. Due to the above it follows $I(a_i) = I(b_i) = I(c_i) = I(d_i) = 1$. We showed above that I is balanced on $\{a_1, b_1, c_1, d_1, \ldots, a_n, b_n, c_n, d_n\}$, that means there exists a $j \in \{1, \ldots, n\}$ such that $I(a_j) + I(b_j) + I(c_j) + I(d_j) < 2$, otherwise we cannot compensate that $I(a_i) = I(b_i) = I(c_i) = I(d_i) = 1$. Because $I(d_j) = 1$ and $I(x_j) = I(a_j)$, $I(y_j) = I(b_j)$, and $I(z_j) = I(c_j)$ it follows that $I(x_j) + I(y_j) + I(z_j) = 0$, which is a contradiction to the fact that $\operatorname{Odd}^3(x_j, y_j, z_j)$ is a clause from φ'. Thus, every balanced solution for φ' restricted to $\operatorname{Var}(\varphi)$ is a solution for φ.

Now let $I : \operatorname{Var}(\varphi) \to \{0,1\}$ be a solution for φ. We prove that exactly one extension of I to $\operatorname{Var}(\varphi')$ is a balanced solution for φ'. It holds $I(x_i) + I(y_i) + I(z_i) = 1$ for every $i \in \{1, \ldots, n\}$, because 1-in-3(x_i, y_i, z_i) is a clause from φ. So, by setting

$$\begin{aligned} I(a_i) &=_{def} I(x_i), \qquad & I(b_i) &=_{def} I(y_i), \\ I(c_i) &=_{def} I(z_i), \qquad & I(d_i) &=_{def} 1 \end{aligned}$$

we extend I such that it is balanced on $\{a_1, b_1, c_1, d_1, \ldots, a_n, b_n, c_n, d_n\}$. If we extend I further according to the above by setting

$$\begin{aligned} I(v') &\neq_{def} I(v), \qquad & I(t_i) &=_{def} 1, \\ I(f_i) &=_{def} 0 \end{aligned}$$

for every $v \in \varphi$ and every $1 \leq i \leq k$, we get a balanced solution for φ'. Due to the above this is the only extension of I to $\mathrm{Var}(\varphi')$ that is a balanced solution for φ'. Hence, there is a one-to-one correspondence between solutions of φ and balanced solutions of φ'.

Obviously φ' can be constructed from φ in logarithmic space, so we showed #CSP(1-in-3) $\leq^{\log}_{!}$ #BAL-CSP(Odd^3), which completes the proof. □

4.3 Hardness Results with unified Proofs

Now we start to look at finite Boolean constraint languages. The first theorem covers all constraint languages that generate IM, IV, IE, or II.

Theorem 4.9. *Let* Γ *be a finite constraint language over* $\{0,1\}$ *such that* $\langle\Gamma\rangle \subseteq$ II *and* $\langle\Gamma\rangle \not\subseteq \mathrm{IN}_2$. *Then* BAL-CSP($\Gamma$) *is* NP*-hard and* #BAL-CSP(Γ) *is* #P*-hard under counting reductions.*

Proof. We show $\mathrm{Imp} \in \langle\Gamma\rangle_{\not\exists,\neq}$, then the hardness for both, the decision and the counting problem, follows from Lemma 4.6 and Proposition 4.2. Since $\langle\Gamma\rangle \not\subseteq \mathrm{IN}_2$ and since $[\neg] = \mathrm{N}_2$ (see Table 2.3), it holds that $\neg$ is no polymorphism of Γ. That means there exists a relation R in Γ such that $\neg \notin \mathrm{Pol}(\Gamma)$. Then there is a $t \in R$ such that $\neg t \notin R$. Since $\langle\Gamma\rangle \subseteq$ II, it holds that c_0 and c_1 are polymorphisms of Γ and in particular of R, which means $c_0(t) = (0,\ldots,0) \in R$ and $c_1(t) = (1,\ldots,1) \in R$. Note that $t \notin \{(0,\ldots,0),(1,\ldots,1)\}$, otherwise $\neg t \in R$. The following equation is true:

$$\mathrm{Imp}(x_0, x_1) = R(x_{t[1]},\ldots,x_{t[\mathrm{arity}(R)]}).$$

Hence, $\mathrm{Imp} \in \langle R\rangle_{\not\exists} \subseteq \langle\Gamma\rangle_{\not\exists}$. Since Imp is irredundant, it follows due to Proposition 3.11 that $\mathrm{Imp} \in \langle\Gamma\rangle_{\not\exists,\neq}$. □

The next theorem deals with constraint languages that generate one of the following co-clones: IM_1, IV_1, IE_1, IS^m_{01}. In the proof we work with weak bases, however we do not need to compute any concrete weak base and we see that weak bases for the above co-clones share some properties.

Theorem 4.10. *Let* Γ *be a finite constraint language over* $\{0,1\}$ *such that* $\mathrm{IM}_1 \subseteq \langle\Gamma\rangle \subsetneq \mathrm{II}_1$. *Then* BAL-CSP($\Gamma$) *is* NP*-hard and* #BAL-CSP(Γ) *is* #P*-hard under counting reductions.*

Proof. Let T-Imp be the relation $\mathrm{C}_1 \times \mathrm{Imp}$. We divide the proof in two parts: in the first part we show T-Imp $\in \langle\Gamma\rangle_{\not\exists,\neq}$. Then we show BAL-CSP(Imp) $\leq^{\log}_{!}$ BAL-CSP(T-Imp) in the second part. The proposition then follows from Lemma 4.6 and Proposition 4.2.

We start with the first part. Let s be a core-size of $\langle\Gamma\rangle$ and let

$$R =_{def} \mathrm{Pol}(\Gamma)(\mathrm{Cols}_s).$$

According to Theorem 3.9 it holds that $\{R\}$ is a weak base of $\langle\Gamma\rangle$ which implies that $\langle R\rangle_{\not\equiv} \subseteq \langle\Gamma\rangle_{\not\equiv}$ due to Corollary 3.5. It is enough to show that T-Imp $\in \langle R\rangle_{\not\equiv}$, then, since T-Imp is irredundant, it follows T-Imp $\in \langle\Gamma\rangle_{\not\equiv,\neq}$ with Proposition 3.11.

We distinguish two cases: $\langle\Gamma\rangle \subseteq \mathrm{IV}_1$ and $\langle\Gamma\rangle \not\subseteq \mathrm{IV}_1$.

Case 1: $\langle\Gamma\rangle \subseteq \mathrm{IV}_1$. Let S be the Boolean relation defined by

$$S(t,x,y) \equiv R(x,\underbrace{y,\ldots,y}_{2^{s-1}-1},\underbrace{t,\ldots,t}_{2^{s-1}}).$$

We show $S =$ T-Imp. Since $\langle\Gamma\rangle \subsetneq \mathrm{II}_1$, it holds that $c_1 \in \mathrm{Pol}(\Gamma)$ and therefore $(1,\ldots,1) \in R$ and $(1,1,1) \in S$. Because $\vee \in \mathrm{V}_1 \subseteq \mathrm{Pol}(\Gamma)$ it follows that the nested application of $\vee$ to all tuples of Cols_s is in R, i.e.,

$$(0,1,\ldots,1) = \mathrm{Cols}_s(1,-) \vee \cdots \vee \mathrm{Cols}_s(s,-) \in R.$$

This means $(1,0,1) \in S$. Since

$$(\underbrace{0,\ldots,0}_{2^{s-1}},\underbrace{1,\ldots,1}_{2^{s-1}}) = \mathrm{Cols}_s(1,-) \in R,$$

it holds that $(1,0,0) \in S$, hence T-Imp $\subseteq S$.

Note that $\mathrm{Pol}(R)$ contains only functions which are monotone and 1-reproducing because $\mathrm{Pol}(R) \subseteq \mathrm{M}_1 = \mathrm{M} \cap \mathrm{R}_1$ (see Table 2.3 and Figure 2.3). Since $\mathrm{Cols}_s(-,2^s) = (1,\ldots,1)$ and since all polymorphisms of Γ are 1-reproducing, it follows $R(-,2^s) = (1,\ldots,1)$ and therefore it holds for all $a,b \in \{0,1\}$ that $(0,a,b) \notin S$.

Finally assume $(1,1,0) \in S$. Then

$$u =_{def} (1,\underbrace{0,\ldots,0}_{2^{s-1}-1},\underbrace{1,\ldots,1}_{2^{s-1}}) \in R.$$

With Lemma 3.8 it follows that there is an s-ary Boolean function $g \in \mathrm{Pol}(\Gamma)$ such that

$$g(\mathrm{Cols}_s(1,-),\ldots,\mathrm{Cols}_s(s,-)) = u.$$

It holds that g is not monotone because $g(0,\ldots,0) = u[1] = 1$ and $g(0,\ldots,0,1) = u[2] = 0$. Since every function from $\mathrm{Pol}(\Gamma)$ is monotone, this is a contradiction. Hence T-Imp $= S$ and therefore T-Imp $\in \langle R\rangle_{\not\equiv} \subseteq \langle\Gamma\rangle_{\not\equiv}$.

Case 2: $\langle\Gamma\rangle \not\subseteq \mathrm{IV}_1$. In this case $\langle\Gamma\rangle = \mathrm{IE}_1$ (see Figure 2.3). Let S be the Boolean relation defined by

$$S(t,x,y) \equiv R(\underbrace{x,\ldots,x}_{2^{s-1}},\underbrace{y,\ldots,y}_{2^{s-1}-1},t).$$

We show $S = \text{T-Imp}$. With the same arguments as in the first case it holds $(1,1,1) \in S$. Because $\wedge \in \mathrm{E}_1$, it follows that

$$(0,\ldots,0,1) = \mathrm{Cols}_s(1,-) \wedge \cdots \wedge \mathrm{Cols}_s(s,-) \in R,$$

which means $(1,0,0) \in S$. Since

$$(\underbrace{0,\ldots,0}_{2^{s-1}},\underbrace{1,\ldots,1}_{2^{s-1}}) = \mathrm{Cols}_s(1,-) \in R,$$

it holds that $(1,0,1) \in S$, hence $\text{T-Imp} \subseteq S$.

Again all functions from $\mathrm{Pol}(\Gamma)$ are monotone and 1-reproducing. So, using the same arguments as in the first case, we have for all $a,b \in \{0,1\}$ that $(0,a,b) \notin S$.

Finally assume $(1,1,0) \in S$. Then

$$u =_{def} (\underbrace{1,\ldots,1}_{2^{s-1}},\underbrace{0,\ldots,0}_{2^{s-1}-1},1) \in R.$$

With Lemma 3.8 it follows that there is an s-ary Boolean function $g \in \mathrm{Pol}(\Gamma)$ such that

$$g(\mathrm{Cols}_s(1,-),\ldots,\mathrm{Cols}_s(s,-)) = u.$$

But g is not monotone because $g(0,\ldots,0) = u[1] = 1$ and $g(1,\ldots,1,0) = u[2^s-1] = 0$. This contradicts that every function of $\mathrm{Pol}(\Gamma)$ is monotone, therefore $(1,1,0) \notin S$. Hence $\text{T-Imp} = S$ and it follows $\text{T-Imp} \in \langle R\rangle_{\not\equiv} \subseteq \langle\Gamma\rangle_{\not\equiv}$.

We showed that in both cases $\text{T-Imp} \in \langle\Gamma\rangle_{\not\equiv}$. Since T-Imp is irredundant it follows with Proposition 3.11 that $\text{T-Imp} \in \langle\Gamma\rangle_{\not\equiv,\neq}$, so due to Proposition 4.2 it holds that $\#\text{BAL-CSP(T-Imp)} \leq^{\log}_{!} \#\text{BAL-CSP}(\Gamma)$.

Now, in the second part of the proof, we show $\#\text{BAL-CSP(Imp)} \leq^{\log}_{!} \#\text{BAL-CSP(T-Imp)}$. Let

$$\varphi =_{def} \mathrm{Imp}(x_1,y_1) \wedge \cdots \wedge \mathrm{Imp}(x_n,y_n)$$

be some Imp-formula. We construct a T-Imp-formula φ' depending on φ. Let t and f be new (not from $\mathrm{Var}(\varphi)$) and distinct variables. We set:

$$\varphi' =_{def} \text{T-Imp}(t, x_1, y_1) \wedge \cdots \wedge \text{T-Imp}(t, x_n, y_n) \wedge \bigwedge_{x \in \mathrm{Var}(\varphi)} \text{T-Imp}(t, f, x).$$

Note that every solution for φ' must map t to 1. Furthermore every balanced solution for φ' must map f to 0 because otherwise 1 is assigned to all variables. Since every balanced solution $I : \mathrm{Var}(\varphi) \to \{0,1\}$ for φ extended by $I(t) =_{def} 1$ and $I(f) =_{def} 0$ is a balanced solution for φ', and conversely every balanced solution for φ' restricted to $\mathrm{Var}(\varphi)$ is a balanced solution for φ, it holds that φ has the same number of balanced solutions as φ'. Therefore and because φ' can be computed in logarithmic space it follows that #BAL-CSP(Imp) $\leq^{\log}_{!}$ #BAL-CSP(T-Imp).

Combining the results from both parts we have

$$\#\text{BAL-CSP(Imp)} \leq^{\log}_{!} \#\text{BAL-CSP(T-Imp)} \leq^{\log}_{!} \#\text{BAL-CSP}(\Gamma).$$

Due to Lemma 4.6, BAL-CSP(Imp) is NP-hard and #BAL-CSP(Imp) is #P-hard, thus BAL-CSP(Γ) is NP-hard and #BAL-CSP(Γ) is #P-hard as well. □

We look at the co-clones IM_2, IV_2, IE_2, and IS^m_{00} for $m \geq 2$ next. The proof for the NP- and #P-hardness is very similar to the previous proof.

Theorem 4.11. *Let Γ be a finite constraint language over $\{0,1\}$ such that $\mathrm{IM}_2 \subseteq \langle\Gamma\rangle \subseteq \mathrm{IV}_2$. Then* BAL-CSP($\Gamma$) *is* NP*-hard and* #BAL-CSP(Γ) *is* #P*-hard under counting reductions.*

Proof. For this proof we define the relation TF-Imp $=_{def}$ $\mathrm{C}_1 \times \mathrm{C}_0 \times$ Imp. Similarly to the proof of Theorem 4.10 we show that TF-Imp $\in \langle\Gamma\rangle_{\nexists,\neq}$ first, and then prove that #BAL-CSP(Imp) $\leq^{\log}_{!}$ #BAL-CSP(TF-Imp).

According to Proposition 2.8 and Figure 2.3 we have $\mathrm{V}_2 \subseteq \mathrm{Pol}(\Gamma) \subsetneq \mathrm{M}$, so due to Table 2.3 it holds that $\vee \in \mathrm{Pol}(\Gamma)$ and every function from Pol(Γ) is monotone.

Let $s \geq 2$ be a core-size of $\langle\Gamma\rangle$. Then $R =_{def} \mathrm{Pol}(\Gamma)(\mathrm{Cols}_s)$ is a weak base for $\langle\Gamma\rangle$ according to Theorem 3.9. Let $n =_{def} 2^s$.

Let S be the Boolean relation defined by

$$S(t,f,x,y) =_{def} R(f, \underbrace{x,\ldots,x}_{\frac{n}{4}-1}, \underbrace{y,\ldots,y}_{\frac{n}{4}}, \underbrace{t,\ldots,t}_{\frac{n}{2}}).$$

We show $S =$ TF-Imp. Since $\mathrm{Cols}_s(1,-) \in R$, it holds that $(1,0,0,0) \in S$ and since $\mathrm{Cols}_s(1,-) \vee \mathrm{Cols}_s(2,-) \in R$ it holds that $(1,0,0,1) \in S$. Because

$$(0,1,\ldots,1) = \mathrm{Cols}_s(1,-) \vee \cdots \vee \mathrm{Cols}_s(s,-) \in R,$$

we have that $(1,0,1,1) \in S$. Thus TF-Imp $\subseteq S$.

Since we have that $\mathrm{Pol}(\Gamma) \subseteq \mathrm{R}_2$, it holds that all polymorphisms of Γ are both 0-reproducing and 1-reproducing. It follows that the first column of R equals $(0,\ldots,0)$ and and the last column of R equals $(1,\ldots,1)$. Therefore it remains to prove $(1,0,1,0) \notin S$. Assume this is not the case. Then

$$u =_{def} (0, \underbrace{1,\ldots,1}_{\frac{n}{4}-1}, \underbrace{0,\ldots,0}_{\frac{n}{4}}, \underbrace{1,\ldots,1}_{\frac{n}{2}}) \in R.$$

Due to Lemma 3.8 there is an s-ary Boolean function $g \in \mathrm{Pol}(\Gamma)$ such that

$$g(\mathrm{Cols}_s(1,-),\ldots,\mathrm{Cols}_s(s,-)) = u.$$

It holds that g is not monotone, because we have that $g(0,\ldots,0,1) = u[2] = 1$ and that $g(0,1,\ldots,1) = u[\frac{n}{2}] = 0$. Since every function from M_2 is monotone this is a contradiction. Hence TF-Imp $= S$ and therefore TF-Imp $\in \langle R \rangle_{\not\equiv} \subseteq \langle \Gamma \rangle_{\not\equiv}$. Note that TF-Imp is irredundant, therefore it follows from Proposition 3.11 that TF-Imp $\in \langle \Gamma \rangle_{\not\equiv,\neq}$. Thus, according to Proposition 4.2 it holds $\#\text{BAL-CSP(TF-Imp)} \leq^{\log}_{!} \#\text{BAL-CSP}(\Gamma)$.

Now we prove $\#\text{BAL-CSP(Imp)} \leq^{\log}_{!} \#\text{BAL-CSP(TF-Imp)}$. Let

$$\varphi =_{def} \mathrm{Imp}(x_1,y_1) \wedge \cdots \wedge \mathrm{Imp}(x_n,y_n)$$

be some {Imp}-formula. We construct a {TF-Imp}-formula φ' depending on φ: let t and f be new and distinct variables. We set:

$$\varphi' =_{def} \text{TF-Imp}(t,f,x_1,y_1) \wedge \cdots \wedge \text{TF-Imp}(t,f,x_n,y_n)$$

Note that the following holds:

$$\varphi' \equiv \varphi \wedge \mathrm{C}_1(t) \wedge \mathrm{C}_0(f).$$

It is easy to see that φ and φ' have the same number of balanced solutions. Therefore and because φ' can be computed in logarithmic space it follows that $\#\text{BAL-CSP(Imp)} \leq^{\log}_{!} \#\text{BAL-CSP(TF-Imp)}$. So together we have

$$\#\text{BAL-CSP(Imp)} \leq^{\log}_{!} \#\text{BAL-CSP(TF-Imp)} \leq^{\log}_{!} \#\text{BAL-CSP}(\Gamma).$$

Lemma 4.6 says that BAL-CSP(Imp) is NP-hard and #BAL-CSP(Imp) is #P-hard, thus BAL-CSP(Γ) is NP-hard and #BAL-CSP(Γ) is #P-hard. □

The following theorem covers the cases IS_0^m and IS_{02}^m for all $m \geq 2$.

Theorem 4.12. *Let Γ be a finite constraint language over $\{0,1\}$ such that $\langle\Gamma\rangle = \mathrm{IS}_{02}^m$ or $\langle\Gamma\rangle = \mathrm{IS}_0^m$ for some natural number $m \geq 2$. Then* BAL-CSP(Γ) *is* NP*-hard and* #BAL-CSP(Γ) *is* #P*-hard under counting reductions.*

Proof. This proof follows the same lines as the proofs for Theorems 4.10 and 4.11.

Let T-Or $=_{def}$ $\mathrm{C}_1 \times \mathrm{Or}^2$ and TF-Or $=_{def}$ $\mathrm{C}_1 \times \mathrm{C}_0 \times \mathrm{Or}^2$. First we look at the cases $\langle\Gamma\rangle = \mathrm{IS}_0^m$ and $\langle\Gamma\rangle = \mathrm{IS}_{02}^m$ separately to show that T-Or $\in \langle\Gamma\rangle_{\not\equiv,\neq}$ in the first case and TF-Or $\in \langle\Gamma\rangle_{\not\equiv,\neq}$ in the second.

Then we prove NP-hardness for BAL-CSP(T-Or) and BAL-CSP(TF-Or) and #P-hardness for #BAL-CSP(T-Or) and #BAL-CSP(TF-Or).

Let s be a core-size of $\langle\Gamma\rangle$ and $R =_{def} \mathrm{Pol}\,\Gamma(\mathrm{Cols}_s)$. Then R is a weak base of $\langle\Gamma\rangle$ due to Theorem 3.9.

Case 1: $\langle\Gamma\rangle = \mathrm{IS}_0^m$ for some $m \geq 2$. We show T-Or $\in \langle\Gamma\rangle_{\not\equiv,\neq}$.

Let S be the Boolean relation defined by

$$S(t,x,y) \equiv R(\underbrace{x,\ldots,x}_{2^{s-1}},\underbrace{y,\ldots,y}_{2^{s-2}},\underbrace{t,\ldots,t}_{2^{s-2}}).$$

Note that $s \geq 2$ according to Table 3.2. We show $S =$ T-Or. Since $\langle\Gamma\rangle \subseteq \mathrm{II}_1$, it holds that $c_1 \in \mathrm{Pol}(\Gamma)$ and therefore $(1,\ldots,1) \in R$ and $(1,1,1) \in S$. Because $\to \in \mathrm{S}_0^m$ it follows that

$$(\underbrace{1,\ldots,1}_{2^{s-1}},\underbrace{0,\ldots,0}_{2^{s-2}},\underbrace{1,\ldots,1}_{2^{s-2}}) = \mathrm{Cols}_s(1,-) \to \mathrm{Cols}_s(2,-) \in R,$$

which means $(1,1,0) \in S$. Since $\mathrm{Cols}_s(1,-) \in R$, it holds that $(1,0,1) \in S$, hence T-Or $\subseteq S$.

Note that $\{\mathrm{dual}(h_m), \to\}$ is a base for S_0^m (see Table 2.3) and both $\mathrm{dual}(h_m)$ and $\to$ are 1-reproducing. Since $\mathrm{Cols}_s(-,2^s) = (1,\ldots,1)$, it follows $R(-,2^s) = (1,\ldots,1)$ and therefore it holds for all $a,b \in \{0,1\}$ that $(0,a,b) \notin S$.

Finally we show that $(1,0,0) \notin S$: Assume $(1,0,0) \in S$. Then according to Lemma 3.8 there is some s-ary function $g \in \mathrm{S}_0^m$ such that

$$(\underbrace{0,\ldots,0}_{2^{s-1}},\underbrace{0,\ldots,0}_{2^{s-2}},\underbrace{1,\ldots,1}_{2^{s-2}}) = g(\mathrm{Cols}_s(1,-),\ldots,\mathrm{Cols}_s(s,-)).$$

That means $g(\mathrm{Cols}_s(-,i)) = 1$ if and only if $2^{s-1}+2^{s-2} < i \leq 2^r$, and therefore we have $g(a_1,\ldots,a_s) = a_1 \wedge a_2$. So it holds that

$$f =_{def} g \circ (pr_1^2, pr_2^2, pr_1^2, \ldots, pr_1^2)$$

is the function represented by $\wedge$ and thus $\wedge \in [\{g\}] \subseteq \mathrm{S}_0^m$. Since $[\{\wedge\}] = \mathrm{E}_2$ it follows $\mathrm{E}_2 \subseteq \mathrm{S}_0^m$, which is not true (see Figure 2.3). Hence, $S =$ T-Or and therefore T-Or $\in \langle R \rangle_{\not\equiv} \subseteq \langle \Gamma \rangle_{\not\equiv}$.

Because T-Or is irredundant, it follows from Proposition 3.11 that T-Or $\in \langle \Gamma \rangle_{\not\equiv,\neq}$. Proposition 4.2 implies #BAL-CSP(T-Or) $\leq_!^{\log}$ #BAL-CSP(Γ).

Case 2: $\langle \Gamma \rangle = \mathrm{IS}_{02}^m$ for some $m \geq 2$. We show TF-Or $\in \langle \Gamma \rangle_{\not\equiv,\neq}$. Let S be the Boolean relation defined by

$$S(t,f,x,y) \equiv R(\underbrace{f,\ldots,f}_{2^{s-2}},\underbrace{x,\ldots,x}_{2^{s-2}},\underbrace{y,\ldots,y}_{2^{s-2}},\underbrace{t,\ldots,t}_{2^{s-2}}).$$

We show $S =$ TF-Or. Since $\mathrm{Cols}_s(1,-) \in R$ and $\mathrm{Cols}_s(2,-) \in R$, it holds that $(1,0,0,1) \in S$ and $(1,0,1,0) \in R$. Note that $\vee \in \mathrm{V}_2 \subseteq \mathrm{Pol}(\Gamma)$ (see Figure 2.3). That means $\mathrm{Cols}_s(1,-) \vee \mathrm{Cols}_s(2,-) \in R$ and therefore $(1,0,1,1) \in S$. Hence, TF-Or $\subseteq S$.

Since $\mathrm{S}_{02}^m \subseteq \mathrm{R}_0$ and $\mathrm{S}_{02}^m \subseteq \mathrm{R}_1$ every polymorphism of Γ is 0-reproducing and 1-reproducing. Therefore the first column of R equals $(0,\ldots,0)$ and the last column of R equals $(1,\ldots,1)$, that means all tuples from S have the form $(1,0,a,b)$ for some $a,b \in \{0,1\}$.

Assume $(1,0,0,0) \in S$. Then according to Proposition 3.11 there is some s-ary function $g \in \mathrm{S}_{02}^m$ such that

$$(\underbrace{0,\ldots,0}_{2^{s-2}},\underbrace{0,\ldots,0}_{2^{s-2}},\underbrace{0,\ldots,0}_{2^{s-2}},\underbrace{1,\ldots,1}_{2^{s-2}}) = g(\mathrm{Cols}_s(1,-),\ldots,\mathrm{Cols}_s(s,-)).$$

In Case 1 we showed that then $E_2 \subseteq \mathrm{Pol}(\Gamma)$, which again is contradiction. Thus, $S =$ TF-Or and therefore TF-Or $\in \langle R \rangle_{\not\equiv} \subseteq \langle \Gamma \rangle_{\not\equiv}$. Since TF-Or is irredundant, it follows that TF-Or $\in \langle \Gamma \rangle_{\not\equiv,\neq}$. With Proposition 4.2 we get #BAL-CSP(TF-Or) $\leq_!^{\log}$ #BAL-CSP(Γ).

To complete the proof it is sufficient to show NP-hardness for the problems BAL-CSP(T-Or) and BAL-CSP(TF-Or), and #P-hardness for the counting versions #BAL-CSP(T-Or) and #BAL-CSP(TF-Or).

We first prove BAL-CSP(Or^2) $\leq_m^{\log}$ BAL-CSP(T-Or) and afterwards show #BAL-CSP(Or^2) $\leq_!^{\log}$ #BAL-CSP(TF-Or). For that let

$$\varphi = \mathrm{Or}^2(x_1,y_1) \wedge \cdots \wedge \mathrm{Or}^2(x_n,y_n)$$

be an $\{\mathrm{Or}^2\}$-formula. We construct a $\{$T-Or$\}$-formula φ' and a $\{$TF-Or$\}$-formula φ'' depending on φ: let t and f be new and distinct variables. We set:

$$\varphi' =_{def} \text{T-Or}(t,x_1,y_1) \wedge \cdots \wedge \text{T-Or}(t,x_n,y_n) \wedge \text{T-Or}(t,t,f),$$

and

$$\varphi'' =_{def} \text{TF-Or}(t, f, x_1, y_1) \wedge \cdots \wedge \text{TF-Or}(t, f, x_n, y_n)$$

Note that φ' and φ'' can be computed in logarithmic space. Obviously every balanced solution $I : \text{Var}(\varphi) \to \{0, 1\}$ for φ can be extended to a balanced solution for φ' and φ'' by setting $I(t) =_{def} 1$ and $I(f) =_{def} 0$. For both φ' and φ'', this is the only satisfying extension of I that is balanced.

Now let $I : \text{Var}(\varphi') \to \{0, 1\}$ be a balanced solution of φ'. It is obvious that $I(t) = 1$. If $I(f) = 0$, the restriction of I to $\text{Var}(\varphi)$ is a balanced solution for φ. If on the other hand $I(f) = 1$, then it holds that I assigns 0 to two variables more of $\text{Var}(\varphi)$ than 1. Let $z \in \text{Var}(\varphi)$ be a variable mapped to 0 by I and let $J : \text{Var}(\varphi) \to \{0, 1\}$ defined by $J(z) = 1$ and for all $x \neq z$: $J(x) = I(x)$. Since every Or^2-clause of φ is satisfied by $I|_{\text{Var}(\varphi)}$ and because of the monotonicity of Or^2, the clauses are satisfied by J as well, so J is a balanced solution for φ. Thus φ has a balanced solution if and only if φ' has one. Therefore we showed $\text{BAL-CSP}(\text{Or}^2) \leq_m^{\log} \text{BAL-CSP}(\text{T-Or})$ which gives us NP-hardness of BAL-CSP(T-Or) due to Lemma 4.7. Note that this reduction is not parsimonious, we will show the #P-hardness of #BAL-CSP(T-Or) later.

Let $I : \text{Var}(\varphi'') \to \{0, 1\}$ be a balanced solution of φ''. Since $I(0) = 0$ and $I(1) = 1$, it follows $I|_{\text{Var}(\varphi)}$ is a balanced solution for φ. So φ has the same number of balanced solutions as φ'' and we showed $\#\text{BAL-CSP}(\text{Or}^2) \leq_!^{\log}$ #BAL-CSP(TF-Or). Thus, due to Lemma 4.7 it holds that BAL-CSP(TF-Or) is NP-hard and #BAL-CSP(TF-Or) is #P-hard.

For the #P-hardness of #BAL-CSP(T-Or) we show that $\#\text{CSP}(\text{Or}^2) \leq_!^{\log}$ #BAL-CSP(T-Or). Let

$$\varphi =_{def} \text{Or}^2(x_1, y_1) \wedge \cdots \wedge \text{Or}^2(x_n, y_n)$$

be an Or^2-formula. Let f, t and v' for every $v \in \text{Var}(\varphi)$ be a new and distinct variables. We define

$$\varphi' =_{def} \text{T-Or}(t, x_1, y_1) \wedge \cdots \wedge \text{T-Or}(t, x_n, y_n) \wedge \bigwedge_{v \in \text{Var}(\varphi)} \text{T-Or}(t, v, v') \wedge \text{T-Or}(t, t, f).$$

The correctness of the reduction can be proven analogously to the correctness of the reduction in the proof of Lemma 4.7. Since φ' can be constructed from φ in logarithmic space, it holds $\#\text{CSP}(\text{Or}^2) \leq_!^{\log}$ #BAL-CSP(T-Or). With Lemma 4.7 it follows #BAL-CSP(T-Or) is hard for #P. This completes the proof. □

4.4 Hardness Results with Non-Unified Poofs

In this section we work in all proofs with concrete irredundant weak bases. The minimal core-sizes provided in Table 3.2 allow us to work with relations of a manageable arity. First we deal with the case where $\langle\Gamma\rangle = \mathrm{ID}_2$.

Theorem 4.13. *Let Γ be a finite constraint language over $\{0,1\}$ such that $\langle\Gamma\rangle = \mathrm{ID}_2$. Then* BAL-CSP($\Gamma$) *is* NP*-hard and* #BAL-CSP(Γ) *is* #P*-hard under counting reductions.*

Proof. Due to Table 2.3 it holds $[h_2] = \mathrm{D}_2$. Note that $h_2(a,b,c)$ is true if and only if $a+b+c \geq 2$. Since 3 is a core-size of ID_2 (see Table 3.2), it follows from Theorem 3.9 that $R =_{def} h_2(\mathrm{Cols}_3)$ is a weak base of ID_2. It can be verified that

$$R = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}$$

Note that the second row is generated by the coordinatewise application of h_2 to the other three rows, which build Cols_3. Clearly, R is irredundant. According to Corollary 3.12 it holds $\langle R\rangle_{\not\equiv,\neq} \subseteq \langle\Gamma\rangle_{\not\equiv,\neq}$.

We define Boolean relations S and T in the following way:

$$S(t,f,x,y) =_{def} R(f,f,x,x,y,y,t,t)$$

and

$$T(t,f,v,w,x,y) =_{def} R(f,f,v,w,y,x,t,t).$$

It follows $\{S,T\} \subseteq \langle R\rangle_{\not\equiv,\neq} \subseteq \langle\Gamma\rangle_{\not\equiv,\neq}$. Therefore, according to Proposition 4.2, it holds #BAL-CSP($\{S,T\}$) $\leq^{\log}_{!}$ #BAL-CSP(Γ).

The following equivalences can be verified:

$$S(t,f,x,y) \equiv \mathrm{C}_1(t) \wedge \mathrm{C}_0(f) \wedge \mathrm{Odd}^2(x,y)$$

$$T(t,f,v,w,x,y) \equiv \mathrm{C}_1(t) \wedge \mathrm{C}_0(f) \wedge \mathrm{Imp}(v,w) \wedge \mathrm{Odd}^2(v,x) \wedge \mathrm{Odd}^2(w,y)$$

We show that #BAL-CSP(Imp) $\leq^{\log}_{!}$ #BAL-CSP($\{S,T\}$). Let

$$\varphi =_{def} \bigwedge_{i=1}^{n} \mathrm{Imp}(x_i,y_i)$$

be an Imp-formula. We construct a $\{S, T\}$-formula: let z', and z'' for every $z \in \mathrm{Var}(\varphi)$ and t and f, be new and distinct variables. We set:

$$\varphi' = \bigwedge_{i=1}^{n} T(t, f, x_i, y_i, x_i', y_i') \wedge S(t, f, x_i', x_i'') \wedge S(t, f, y_i', y_i'').$$

It holds that

$$\varphi' \equiv \varphi \wedge \bigwedge_{i=1}^{n} \mathrm{Imp}(x_i, y_i) \wedge \bigwedge_{z \in \varphi} \mathrm{Odd}^2(z, z') \wedge \mathrm{Odd}^2(z', z'') \wedge \mathrm{C}_1(t) \wedge \mathrm{C}_0(f)$$

Every balanced solution $I : \mathrm{Var}(\varphi') \to \{0, 1\}$ of φ' is already balanced on $\mathrm{Var}(\varphi') \setminus \mathrm{Var}(\varphi)$, because $I(t) \neq I(f)$ and because the clauses of the form $\mathrm{Odd}^2(z', z'')$ provide that for every $z \in \varphi$ it holds $I(z') \neq I(z'')$. It follows that I is balanced on $\mathrm{Var}(\varphi)$ as well and therefore $I|_{\mathrm{Var}(\varphi)}$ is a balanced solution for φ.

Conversely, every balanced solution $I : \mathrm{Var}(\varphi) \to \{0, 1\}$ for φ can be extended to a balanced solution of φ by setting

$$\begin{array}{ll} I(t) =_{def} 1, & I(f) =_{def} 0, \\ I(z') \neq_{def} I(z), & I(z'') =_{def} I(z) \quad \text{for every } z \in \mathrm{Var}(\varphi). \end{array}$$

Since every other extension of I to $\mathrm{Var}(\varphi')$ does not satisfy φ', we have a one-to-one correspondence between balanced solutions of φ and the balanced solutions of φ'. Hence,

$$\#\text{BAL-CSP}(\mathrm{Imp}) \leq_!^{\log} \#\text{BAL-CSP}(\{S, T\}) \leq_!^{\log} \#\text{BAL-CSP}(\Gamma).$$

Due to Lemma 4.6 #BAL-CSP(Imp) is hard for #P and BAL-CSP(Imp) is hard for NP. Therefore #BAL-CSP(Γ) is hard for #P and BAL-CSP(Γ) is hard for NP. □

With the next four theorems we cover the cases IL, IL_1, IL_2 and IL_3. Note that the hardness-results transfer to IL_0 due to Proposition 4.3, because $\mathrm{IL}_0 = \mathrm{dual}(\mathrm{IL}_1)$. The proofs of the following theorems are very similar, however they differ in so many details that there does not seem to be a way to combine them.

Theorem 4.14. *Let Γ be a finite constraint language over $\{0, 1\}$ such that $\langle\Gamma\rangle = \mathrm{IL}$. Then* BAL-CSP($\Gamma$) *is* NP*-hard and* #BAL-CSP(Γ) *is* #P*-hard under counting reductions.*

Proof. The minimal core-size of IL is 2 (see Table 3.2), therefore $R =_{def} \mathrm{L}(\mathrm{Cols}_2)$ is weak base of IL. It can be verified that

$$R = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} = \mathrm{Even}^4.$$

The second and the third row are from Cols_2, the other ones result from the application of functions from L to them.

Since Even^4 is obviously irredundant it follows from Corollary 3.12 that $\mathrm{Even}^4 \in \langle\Gamma\rangle_{\not\equiv,\neq}$. Therefore we know $\#\text{BAL-CSP}(\mathrm{Even}^4) \leq_!^{\log} \#\text{BAL-CSP}(\Gamma)$ due to Proposition 4.2.

We show $\#\text{BAL-CSP}(\mathrm{Odd}^3) \leq_c^{\log} \#\text{BAL-CSP}(\mathrm{Even}^4)$. Let

$$\varphi =_{def} \bigwedge_{i=1}^{n} \mathrm{Odd}^3(x_i, y_i, z_i)$$

be an Odd^3-formula. Let $k =_{def} |\operatorname{Var}(\varphi)|$ and let $t, t_1, \ldots, t_k, f, f_1, \ldots, f_k$ be new and distinct variables. We set:

$$\varphi' =_{def} \bigwedge_{i=1}^{n} \mathrm{Even}^4(t, x_i, y_i, z_i) \wedge \bigwedge_{i_1}^{k} \mathrm{Even}^4(t, t, t, t_i) \wedge \mathrm{Even}^4(f, f, f, f_i).$$

We prove that φ' has exactly twice as many balanced solutions as φ. First note that the clauses $\mathrm{Even}^4(t, t, t, t_i)$ and $\mathrm{Even}^4(f, f, f, f_i)$ for all $i \in \{1, \ldots, k\}$ imply that every solution of φ' maps $t_1, \ldots, t_k$ to the same value as t and $f_1, \ldots, f_k$ to the same value as f. Further, every balanced solution of φ' must map f and t to different values, because otherwise this value would be assigned to at least $2k + 2$ variables, which is more than half of the variables of φ.

Now let $I : \operatorname{Var}(\varphi) \to \{0, 1\}$ be a balanced solution of φ. It can be verified that we get a balanced solution for φ' by setting

$$\begin{aligned} &I(f) =_{def} I(f_1) =_{def} \ldots =_{def} I(f_k) =_{def} 0 \quad \text{and} \\ &I(t) =_{def} I(t_1) =_{def} \ldots =_{def} I(t_k) =_{def} 1. \end{aligned}$$

This is the only extension of I to $\operatorname{Var}(\varphi')$ that is balanced and satisfies φ', because if we map t to 0, then the $I(t) + I(x_i) + I(y_i) + I(z_i)$ would be odd for every $i \in \{1, \ldots, n\}$.

Finally let $I : \mathrm{Var}(\varphi') \to \{0,1\}$ be a balanced solution for φ. Since I is already balanced on the new introduced variables, it holds that I is also balanced on $\mathrm{Var}(\varphi)$. For every $1 \leq i \leq n$ we know that $I(t)+I(x_i)+I(y_i)+I(z_i)$ is even, therefore $I(x_i)+I(y_i)+I(z_i)$ is odd if and only if $I(t) = 0$. That means $I|_{\mathrm{Var}(\varphi)}$ is a balanced solution for φ if and only if $I(t) = 1$.

Since $\neg \in \mathrm{N}_2 \subseteq \mathrm{L}$, it holds that I', defined by $I'(x) = \neg I(x)$, is a balanced solution of φ' as well. That means exactly half of that balanced solutions of φ' can be restricted to a balanced solution of φ. Thus, φ' has exactly twice as many balanced solutions as φ. Because φ' can be computed in logarithmic space, it follows that we have a counting reduction between #BAL-CSP(Odd^3) and #BAL-CSP(Even^4). Since #BAL-CSP(Odd^3) is #P-hard and BAL-CSP(Odd^3) is NP-hard due to Lemma 4.8, and since we showed #BAL-CSP(Even^4) $\leq_!^{\log}$ #BAL-CSP(Γ) above, it holds that #BAL-CSP(Γ) is #P-hard under counting reductions and BAL-CSP(Γ) is NP-hard. □

Theorem 4.15. *Let Γ be a finite constraint language over $\{0,1\}$ such that $\langle\Gamma\rangle = \mathrm{IL}_3$. Then* BAL-CSP($\Gamma$) *is* NP*-hard and* #BAL-CSP(Γ) *is* #P*-hard under counting reductions.*

Proof. The minimal core-size of IL_3 is 3 due to Table 3.2, therefore it follows from Theorem 3.9 that $R =_{def} \mathrm{L}_3(\mathrm{Cols}_3)$ is a weak base for IL_3. It can be verified that

$$R = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Note that the first three tuples of R form Cols_3. It holds that

$$R_4(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8) \equiv$$

$$\mathrm{Even}^4(x_1,x_2,x_3,x_4) \wedge \mathrm{Odd}^2(x_1,x_8) \wedge \mathrm{Odd}^2(x_2,x_7) \wedge \mathrm{Odd}^2(x_3,x_6) \wedge \mathrm{Odd}^2(x_4,x_5).$$

It can be seen that R is irredundant, therefore it follows from Corollary 3.12 that $R \in \langle\Gamma\rangle_{\not\equiv,\neq}$. So we know #BAL-CSP($R$) $\leq_!^{\log}$ #BAL-CSP(Γ) due to Proposition 4.2.

In the proof of Theorem 4.14 we showed that BAL-CSP(Even^4) is NP-hard and #BAL-CSP(Even^4) is #P-hard under counting reductions. Therefore it

suffices to show #BAL-CSP(Even^4) $\leq^{\log}_{!}$ #BAL-CSP(R) to prove this theorem. Let

$$\varphi =_{def} \bigwedge_{i=1}^{n} \mathrm{Even}^4(w_i, x_i, y_i, z_i)$$

be an Even^4-formula. Without loss of generality assume that $k =_{def} |\mathrm{Var}(\varphi)|$ is at least 2. For every $v \in \mathrm{Var}(\varphi)$ let $v^1, \ldots, v^k$ be new and distinct variables. We set:

$$\varphi' =_{def} \bigwedge_{i=1}^{n} R(w_i, x_i, y_i, z_i, z_i^1, y_i^1, x_i^1, w_i^1) \wedge \cdots \wedge R(w_i, x_i, y_i, z_i, z_i^k, y_i^k, x_i^k, w_i^k).$$

Then the following equivalence holds:

$$\varphi' \equiv \varphi \wedge \bigwedge_{v \in \mathrm{Var}(\varphi)} \mathrm{Odd}^2(v, v^1) \wedge \cdots \wedge \mathrm{Odd}^2(v, v^k).$$

Let $I : \mathrm{Var}(\varphi') \to \{0,1\}$ be a balanced solution of φ'. Clearly $I|_{\mathrm{Var}(\varphi)}$ is a solution of φ. We show that $I|_{\mathrm{Var}(\varphi)}$ is balanced. Let l_0 be the number of variables mapped to 0 and l_1 be the number of variables mapped to 1 by $I|_{\mathrm{Var}(\varphi)}$. It holds for every $v \in \mathrm{Var}(\varphi)$ and for every $1 \leq i \leq k$ that $I(v) \neq I(v^i)$ in order to satisfy $\mathrm{Odd}^2(v, v^i)$. Therefore I maps $l_0 + kl_1$ variables to 0 and $l_1 + kl_0$ variables to 1. Since I is balanced, it follows $l_0 + kl_1 = l_1 + kl_0$, which implies $l_0 = l_1$ because $k \geq 2$. Hence, $I|_{\mathrm{Var}(\varphi)}$ is a balanced solution for φ.

Obviously I is uniquely determined by $I|_{\mathrm{Var}(\varphi)}$. Now let $I : \mathrm{Var}(\varphi) \to \{0,1\}$ be a balanced solution for φ. For every $i \in \{1, \ldots, k\}$ and ever $v \in \mathrm{Var}(\varphi)$ we set $I(v^i) \neq_{def} I(v)$. Clearly this extension satisfies φ' and with the above equations it can be seen that it is balanced on $\mathrm{Var}(\varphi')$.

Thus we have a one-to-one correspondence between balanced solutions from φ and balanced solutions from φ'. Since φ' can be computed in logarithmic space, it follows #BAL-CSP(Even^4) $\leq^{\log}_{!}$ #BAL-CSP(R), which completes the proof. □

Theorem 4.16. *Let Γ be a finite constraint language over $\{0,1\}$ such that $\langle\Gamma\rangle = \mathrm{IL}_1$. Then* BAL-CSP($\Gamma$) *is* NP*-hard and* #BAL-CSP(Γ) *is* #P*-hard under counting reductions.*

Proof. Since 2 is a core-size of IL_1 due to Table 3.2 it follows with Theorem 3.9 that $R =_{def} \mathrm{L}_1(\mathrm{Cols}_2)$ is a weak base for IL_1. It can be verified that

$$R = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$

Note that the first two rows of R are the ones from Cols_2. It is easy to see that $R = \mathrm{Odd}^2 \times \mathrm{C}_1$ and that R is irredundant. It follows from Corollary 3.12 that $R \in \langle\Gamma\rangle_{\not\equiv,\neq}$. Therefore we know $\#\text{BAL-CSP}(R) \leq_!^{\log} \#\text{BAL-CSP}(\Gamma)$ due to Proposition 4.2.

We show $\#\text{BAL-CSP}(\mathrm{Odd}^3) \leq_c^{\log} \#\text{BAL-CSP}(R)$, then the theorem follows with Lemma 4.8. Let

$$\bigwedge_{i=1}^{n} \mathrm{Odd}^3(x_i, y_i, z_i)$$

be an Odd^3-formula. Let $k =_{def} \mathrm{Var}(\varphi)$ and let $t, t_1, \ldots, t_k, f, f_1, \ldots, f_k$ be new and distinct variables. We set:

$$\varphi' =_{def} \bigwedge_{i=1}^{n} R(x_i, y_i, z_i, t) \wedge \bigwedge_{i_1}^{k} \mathrm{Even}^4(f_i, f_i, t_i, t_i)$$

The correctness of the reduction can be shown with similar arguments as in the proof of Theorem 4.14. Note that here we get a parsimonious reduction because the variable t must be mapped to 1 by every solution.

Since φ' can be constructed in logarithmic space, the proof is complete. □

Theorem 4.17. *Let Γ be a finite constraint language over $\{0,1\}$ such that $\langle\Gamma\rangle = \mathrm{IL}_2$. Then* BAL-CSP($\Gamma$) *is* NP*-hard and* #BAL-CSP(Γ) *is* #P*-hard under counting reductions.*

Proof. The minimal core-size of IL_2 is 3 (see Table 3.2), therefore $R =_{def} \mathrm{L}_2(\mathrm{Cols}_3)$ is a weak base for IL_2 according to Theorem 3.9. It holds that

$$R = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

The first three tuples of R are the ones from Cols_3. It can be verified that the following equivalence is true:

$$R(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) \equiv$$

$$\mathrm{Odd}^3(x_2, x_3, x_4) \wedge \mathrm{Odd}^2(x_2, x_7) \wedge \mathrm{Odd}^2(x_3, x_6) \wedge \mathrm{Odd}^2(x_4, x_5) \wedge \mathrm{C}_0(x_1) \wedge \mathrm{C}_1(x_8)$$

It is easy to see that R is irredundant. So it follows from Corollary 3.12 that $R \in \langle\Gamma\rangle_{\not\equiv,\neq}$. Therefore we know $\#\text{BAL-CSP}(R) \leq_!^{\log} \#\text{BAL-CSP}(\Gamma)$ due to Proposition 4.2.

Lemma 4.8 says that BAL-CSP(Odd^3) is NP-hard and #BAL-CSP(Odd^3) is #P-hard under parsimonious reductions. Therefore to prove this theorem it suffices to show $\#\text{BAL-CSP}(\mathrm{Odd}^3) \leq_!^{\log} \#\text{BAL-CSP}(R)$.

Let

$$\varphi =_{def} \bigwedge_{i=1}^{n} \mathrm{Odd}^3(x_i, y_i, z_i)$$

be an Odd^3-formula. Without loss of generality assume that $k =_{def} |\mathrm{Var}(\varphi)|$ is at least 2. Let f, t and $v^1, \ldots, v^k$ for every $v \in \mathrm{Var}(\varphi)$ be new and distinct variables. We set:

$$\varphi' =_{def} \bigwedge_{i=1}^{n} R(t, x_i, y_i, z_i, z_i^1, y_i^1, x_i^1, f) \wedge \cdots \wedge R(t, x_i, y_i, z_i, z_i^k, y_i^k, x_i^k, f).$$

Then the following equivalence holds:

$$\varphi' \equiv \varphi \wedge \mathrm{C}_0(f) \wedge \mathrm{C}_1(t) \wedge \bigwedge_{v \in \mathrm{Var}(\varphi)} \mathrm{Odd}^2(v, v^1) \wedge \cdots \wedge \mathrm{Odd}^2(v, v^k)$$

The correctness of this reduction can be proven with using similar arguments as in the proof of Theorem 4.15.

Since φ' can be constructed in logarithmic space from φ, it follows that $\#\text{BAL-CSP}(\mathrm{Odd}^3) \leq_!^{\log} \#\text{BAL-CSP}(R)$, which completes the proof. □

Finally we prove NP-hardness and #P-hardness for the non-Schaefer constraint languages. Note that the case II is already covered in Theorem 4.9 and that II_0 is dual to II_1. For the remaining four non-Schaefer co-clones the proofs again are very similar, but not obviously unifiable.

Theorem 4.18. *Let Γ be a finite constraint language over $\{0,1\}$ such that $\langle\Gamma\rangle = \mathrm{II}_2$. Then* BAL-CSP($\Gamma$) *is* NP*-complete and* #BAL-CSP(Γ) *is complete for* #P *under counting reductions.*

Proof. Due to Table 3.2, the minimal core-size of II_2 is 3, therefore $R =_{def} \mathrm{I}_2(\mathrm{Cols}_3)$ is a weak base for II_2 according to Theorem 3.9. Because $[id] = \mathrm{I}_2$ (see Table 2.3), it holds that $R = \mathrm{Cols}_3$. That means

$$R = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

It can be verified that the following equivalence is true:

$$\begin{aligned} R(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) \equiv & \text{1-in-3}(x_2, x_3, x_5) \wedge \mathrm{C}_0(x_1) \wedge \mathrm{C}_1(x_8) \\ & \wedge \mathrm{Odd}^2(x_2, x_7) \wedge \mathrm{Odd}^2(x_3, x_6) \wedge \mathrm{Odd}^2(x_4, x_5) \end{aligned}$$

Since R is irredundant, it follows $R \in \langle\Gamma\rangle_{\not\equiv,\neq}$ from Corollary 3.12. Therefore we have #BAL-CSP(R) $\leq^{\log}_{!}$ #BAL-CSP(Γ) due to Proposition 4.2.

It can be seen that 1-in-3 generates the co-clone II_2 in Table 3.2, therefore it follows from Schaefer's Theorem (Theorem 2.11) that CSP(1-in-3) is NP-hard and from Theorem 2.12 that #CSP(1-in-3) is hard for #P under parsimonious reductions. Hence, showing #CSP(1-in-3) $\leq^{\log}_{!}$ #BAL-CSP(R) completes the proof. Let

$$\varphi =_{def} \bigwedge_{i=1}^{n} \text{1-in-3}(x_i, y_i, z_i)$$

be a constraint formula over {1-in-3}. Let f, t and v' for every $v \in \mathrm{Var}(\varphi)$ be new and distinct variables. We define an R-formula φ':

$$\varphi' =_{def} \bigwedge_{i=1}^{n} R(f, x_i, y_i, z_i', z_i, y_i', x_i', t)$$

According to the above it holds

$$\varphi' \equiv \varphi \wedge \bigwedge_{v \in \mathrm{Var}(\varphi)} \mathrm{Odd}^2(v_i, v_i') \wedge \mathrm{C}_0(f) \wedge \mathrm{C}_1(t).$$

Obviously every balanced solution of φ' satisfies φ as well and every solution of varphi can be extended uniquely to a balanced solution for φ'. Since the formula φ' can be constructed in logarithmic space it follows that #CSP(1-in-3) $\leq^{\log}_{!}$ #BAL-CSP(R). □

Theorem 4.19. *Let Γ be a finite constraint language over $\{0,1\}$ such that $\langle\Gamma\rangle = \mathrm{IN}_2$. Then* BAL-CSP($\Gamma$) *is* NP*-complete and* #BAL-CSP(Γ) *is complete for* #P *under counting reductions.*

Proof. Due to Table 3.2, the minimal core-size of IN_2 is 3, therefore $R =_{def} \mathrm{N}_2(\mathrm{Cols}_3)$ is a weak base for IN_2 according to Theorem 3.9. It holds that

$$R = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

It can be verified that $R = \mathrm{Cols}_3 \cup \mathrm{dual}(\mathrm{Cols}_3)$. Since R is irredundant, it follows from Corollary 3.12 that $R \in \langle\Gamma\rangle_{\not\equiv,\neq}$. Therefore we have #BAL-CSP(R) $\leq^{\log}_{!}$ #BAL-CSP(Γ) due to Proposition 4.2.

Due to Table 3.2 it holds that 1-in-3 generates the II_2, therefore it follows from Schaefer's Theorem (Theorem 2.11) that CSP(1-in-3) is NP-hard and from Theorem 2.11 that #CSP(1-in-3) is hard for #P under parsimonious reductions. Hence, showing #CSP(1-in-3) $\leq_c^{\log}$ #BAL-CSP(R) completes the proof. Let

$$\varphi =_{def} \bigwedge_{i=1}^{n} \text{1-in-3}(x_i, y_i, z_i)$$

be a constraint formula over {1-in-3}. Let f, t and v' for every $v \in \mathrm{Var}(\varphi)$ be new and distinct variables. We define an R-formula φ':

$$\psi =_{def} \bigwedge_{i=1}^{n} R(f, x_i, y_i, z_i', z_i, y_i', x_i', t)$$

We show that the number of balanced solutions for ψ is exactly twice the number of solutions for φ. Let φ' be the formula from φ in the proof of Theorem 4.18. Note that a solution for ψ maps f to 0 if and only if it is a solution for φ'. It maps f to 1 if and only if it is a solution for dual(φ').

Since the number of solutions for φ and balanced solutions for φ' is exactly the same (see proof of Theorem 4.18), and since φ' has exactly the same number of balanced solutions as dual(φ), it holds that the number of balanced solutions for ψ is exactly twice the number of solutions for φ.

Since the formula ψ can be constructed in logarithmic space it follows that #CSP(1-in-3) $\leq_c^{\log}$ #BAL-CSP(R). □

Theorem 4.20. *Let Γ be a finite constraint language over $\{0,1\}$ such that $\langle\Gamma\rangle = \mathrm{II}_0$. Then* BAL-CSP($\Gamma$) *is* NP*-complete and* #BAL-CSP(Γ) *is complete for* #P *under counting reductions.*

Proof. Due to Table 3.2, the minimal core-size of II_0 is 2, therefore $R =_{def} \mathrm{I}_0(\mathrm{Cols}_2)$ is a weak base for II_0 according to Theorem 3.9. It holds that

$$R = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}.$$

Since R is irredundant, it follows $R \in \langle\Gamma\rangle_{\not\equiv,\neq}$ from Corollary 3.12. Therefore we have #BAL-CSP(R) $\leq_!^{\log}$ #BAL-CSP(Γ) due to Proposition 4.2.

Since $\langle R\rangle = \mathrm{II}_0$ and $\langle \mathrm{C}_1\rangle = \mathrm{IR}_1$, it holds that

$$\langle\{R, \mathrm{C}_1\}\rangle = \langle \mathrm{II}_0 \cup \mathrm{IR}_1\rangle = \mathrm{II}_2.$$

Therefore it follows Theorem 2.11 that $\mathrm{CSP}(\{R, \mathrm{C}_1\})$ is NP-hard and from Theorem 2.12 that $\#\mathrm{CSP}(\{R, \mathrm{C}_1\})$ is hard for #P under parsimonious reductions. Hence, showing $\#\mathrm{CSP}(\{R, \mathrm{C}_1\}) \leq^{\log}_{!} \#\text{BAL-CSP}(R)$ completes the proof. Let

$$\varphi =_{def} \bigwedge_{i=1}^{n} R(w_i, x_i, y_i, z_i) \wedge \bigwedge_{i=1}^{m} \mathrm{C}_1(v_i)$$

be a constraint formula over $\{R, \mathrm{C}_1\}$. Let $k =_{def} |\operatorname{Var}(\varphi)|$ the number of variables appearing in φ and let $f, f_1, \ldots, f_k, t, t_1, \ldots, t_k$ and x' for every $x \in \operatorname{Var}(\varphi)$ be new and distinct variables. We define an R-formula φ':

$$\begin{aligned}\varphi' =_{def} \bigwedge_{i=1}^{n} R(w_i, x_i, y_i, z_i) \wedge \bigwedge_{x \in \operatorname{Var}(\varphi)} R(f, x, x', t) \\ \wedge \bigwedge_{i=1}^{k} R(f_i, f_i, t, t_i) \wedge \bigwedge_{i=1}^{m} R(f, f, t, v_i)\end{aligned}$$

Note that $|\operatorname{Var}(\varphi')| = 4k + 2$. Let $I : \operatorname{Var}(\varphi') \to \{0, 1\}$ be a balanced solution for φ'. The clauses of the form $R(f_i, f_i, t, t_i)$ give us that $I(f_1) = \cdots = I(f_k) = 0$ and $I(t) = I(t_1) = \cdots = I(t_k)$. Since f appears in some of the clauses in the first position, we have $I(f) = 0$. It follows $I(t) = I(t_1) = \cdots = I(t_k) = 1$, otherwise I would map at least $2k+2$ variables to 0 which means that I would not be balanced. Because of the clauses of the type $R(f, f, t, v_i)$, it holds that $I(v_1) = \cdots = I(v_m) = I(t) = 1$, therefore I satisfies φ.

From the clauses of the type $R(f, x, x', t)$ it follows that for all $x \in \operatorname{Var}(\varphi)$ it holds $I(x) \neq I(x')$. That means I is uniquely determined by $I|_{\operatorname{Var}(\varphi)}$. So for every solution of φ we can define at most one extension that is a balanced solution for φ.

Now let $I : \operatorname{Var}(\varphi) \to \{0, 1\}$. We extend I by defining:

$$\begin{array}{lll} I(f) & =_{def}\ I(f_1) =_{def} \ldots =_{def} I(f_k) =_{def} 0, & \\ I(t) & =_{def}\ I(t_1) =_{def} \ldots =_{def} I(t_k) =_{def} 1, & \text{and} \\ I(x') & \neq_{def}\ I(x) & \text{for every } x \in \operatorname{Var}(\varphi). \end{array}$$

Obviously this extension is balanced and, since for all $i \in 1, \ldots, m$ we have $I(v_i) = 1$, it satisfies φ'. Hence, the balanced solutions for φ' are exactly unique extensions of solutions for φ. Since φ' can be constructed in logarithmic space, we proved $\#\mathrm{CSP}(\{R, \mathrm{C}_1\}) \leq^{\log}_{!} \#\text{BAL-CSP}(R)$. □

Theorem 4.21. *Let Γ be a finite constraint language over $\{0, 1\}$ such that $\langle\Gamma\rangle = \mathrm{II}_0$. Then* BAL-CSP($\Gamma$) *is* NP*-complete and* #BAL-CSP(Γ) *is complete for* #P *under counting reductions.*

Proof. Due to Table 3.2, the minimal core-size of IN is 2, therefore $R =_{def} \mathrm{N}(\mathrm{Cols}_2)$ is a weak base for II_0 according to Theorem 3.9. It holds that

$$R = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$

Since R is irredundant, it follows $R \in \langle\Gamma\rangle_{\not\equiv,\neq}$ from Corollary 3.12. Therefore we have $\#\mathrm{BAL\text{-}CSP}(R) \leq_!^{\log} \#\mathrm{BAL\text{-}CSP}(\Gamma)$ due to Proposition 4.2.

Since $\langle R\rangle = \mathrm{II}_0$ and $\langle \mathrm{C}_0, \mathrm{C}_1\rangle = \mathrm{IR}_2$, it holds that

$$\langle\{R, \mathrm{C}_0, \mathrm{C}_1\}\rangle = \langle \mathrm{IN} \cup \mathrm{IR}_2\rangle = \mathrm{II}_2.$$

Therefore it follows from Theorem 2.11 that $\mathrm{CSP}(\{R, \mathrm{C}_0, \mathrm{C}_1\})$ is NP-hard and from Theorem 2.12 that $\#\mathrm{CSP}(\{R, \mathrm{C}_0, \mathrm{C}_1\})$ is hard for #P under parsimonious reductions. Hence, showing $\#\mathrm{CSP}(\{R, \mathrm{C}_0, \mathrm{C}_1\}) \leq_!^{\log} \#\mathrm{BAL\text{-}CSP}(R)$ completes the proof. Let

$$\varphi =_{def} \bigwedge_{i=1}^{n} R(w_i, x_i, y_i, z_i) \wedge \bigwedge_{i=1}^{m_0} \mathrm{C}_0(v_i) \wedge \bigwedge_{i=1}^{m_1} \mathrm{C}_1(u_i)$$

be a constraint formula over $\{R, \mathrm{C}_0, \mathrm{C}_1\}$. Let $k =_{def} |\mathrm{Var}(\varphi)|$ be the number of variables occurring in φ and let $f, f_1, \ldots, f_k, t, t_1, \ldots, t_k$ and x' for every $x \in \mathrm{Var}(\varphi)$ be and distinct variables. We define an R-formula φ':

$$\begin{aligned} \varphi' =_{def} & \bigwedge_{i=1}^{n} R(w_i, x_i, y_i, z_i) \wedge \bigwedge_{x \in \mathrm{Var}(\varphi)} R(f, x, x', t) \\ & \wedge \bigwedge_{i=1}^{k} R(f, f, f, f_i) \wedge R(t, t, t, t_i) \\ & \wedge \bigwedge_{i=1}^{m_0} R(f, f, f, v_i) \wedge \bigwedge_{i=1}^{m_1} R(t, t, t, u_i). \end{aligned}$$

This construction is very similar to the one in the proof of Theorem 4.20. The correctness of this reduction can be shown with analogous arguments. Note that R is complementive, therefore we get here that the number of balanced solutions for φ' is exactly twice the number of solutions for φ. Since φ' can be constructed in logarithmic time, this gives $\#\mathrm{CSP}(\{R, \mathrm{C}_0, \mathrm{C}_1\}) \leq_c^{\log} \#\mathrm{BAL\text{-}CSP}(R)$. □

Our main theorem summarizes the previous results to state a complete complexity classification for the balanced satisfiability problems. A graphical overview of this dichotomy can be seen in Figure 4.4.

Theorem 4.22. *Let* Γ *be a finite constraint language over* $\{0,1\}$.

- BAL-CSP(Γ) *is decidable in polynomial time if* $\langle\Gamma\rangle \subseteq \mathrm{ID}_1$, *and in all other cases* BAL-CSP(Γ) *is* NP*-complete.*
- #BAL-CSP(Γ) *is computable in polynomial time if* $\langle\Gamma\rangle \subseteq \mathrm{ID}_1$, *and in all other cases* #BAL-CSP(Γ) *is* #P*-complete under counting reductions.*

Proof. The upper complexity bounds follow from Proposition 4.1 and Theorem 4.5. The lower bounds follow from Theorems 4.9-4.21 and Proposition 4.3. □

4.5 Exact Satisfiability

We have shown that weak bases can be used to obtain a full classification for the balanced satisfiability problem and its counting version. It turns out, that the complexity of these problems depends only on the co-clone generated by the according constraint language, although in the first place we could only prove that it depends only on the generated weak system without equality (see Proposition 4.2). In some cases we had to work with concrete weak bases (Section 4.4) and deal with every co-clone one-by-one, and in some cases we could work with properties of weak bases and deal with several co-clones at the same time (Section 4.3).

Note that the classification for balanced satisfiability also holds for the more general problem of exact satisfiability, also known in the literature as K-ONES.

Problem:	K-ONES(Γ)
Input:	a Γ-formula φ, $k \in \mathbb{N}$
Question:	does φ have a solution that maps exactly k variables to 1?

Since every Boolean constraint formula φ has a balanced solution if and only if φ has a solution with exactly $\frac{1}{2}|\operatorname{Var}(\varphi)|$ variables set to 1, it holds that BAL-CSP(Γ) reduces trivially to K-ONES(Γ) for every Boolean constraint language Γ. The proof for Theorem 4.5, which covers all polynomial time cases of balanced satisfiability, gives a polynomial time result for the exact satisfiability as well. The results for #BAL-CSP(Γ) transfer to the counting variant #K-ONES(Γ) of the exact satisfiability problem in the same way, therefore we obtain the following classification.

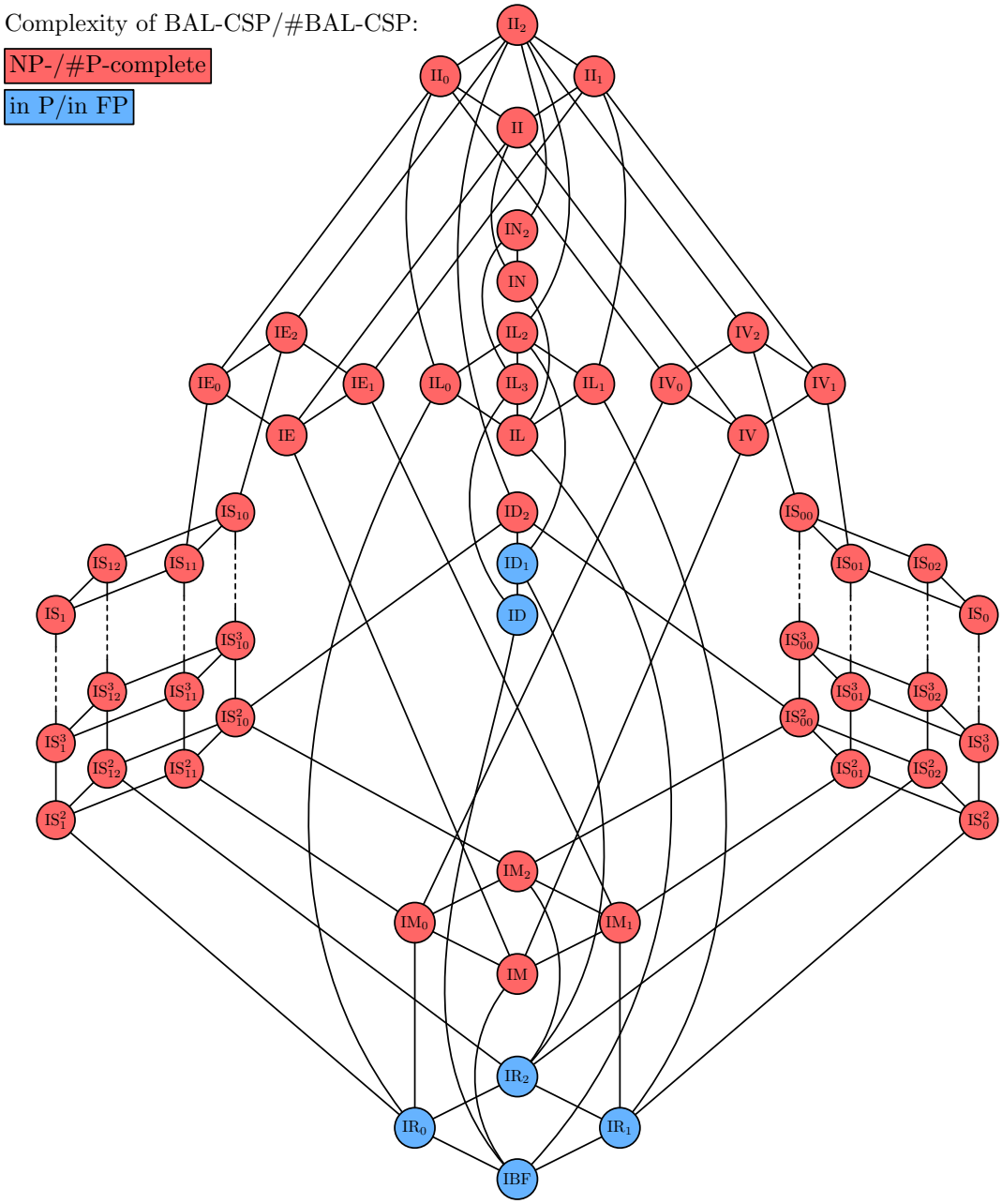

Figure 4.1: The Complexity of Balanced Satisfiability

Corollary 4.23. *Let* Γ *be a finite constraint language over* $\{0,1\}$.

- K-ONES(Γ) *is decidable in polynomial time if* $\langle\Gamma\rangle \subseteq \mathrm{ID}_1$, *and in all other cases* K-ONES(Γ) *is* NP*-complete.*
- #K-ONES(Γ) *is computable in polynomial time if* $\langle\Gamma\rangle \subseteq \mathrm{ID}_1$, *and in all other cases* #K-ONES(Γ) *is* #P*-complete.*

Chapter 5

Default Logic

As a second example for applications of weak bases we classify the complexity of constraint problems in the context of propositional default logic. Default logic is a nonmonotonic logic introduced by Reiter in [Rei80]. It allows to model human behavior to complement incomplete information with default assumptions. We give an often used example: if someone told us that there was a bird in the garden, then we would intuitively think about a bird that can fly. However, there are birds like ostriches and penguins, that cannot fly, and nobody told us that the special bird in the garden is a flying bird. In default logic we can express as a rule that we assume by default every bird to be able to fly, as long as we do not know that the bird belongs to a race of non-flying birds. Due to the ability to express such default assumptions, default logic is important in artificial intelligence.

We look at three computational problems that arise in this context: the first one is the question if a given set of facts and a given set of default rules are consistent; the second is whether the default rules can be applied such that an additionally given formula results as a consequence; and the last one is the question whether a given formula results as a consequence, no matter in which order the rules are applied. The last two questions are known as *credulous reasoning* and *skeptical reasoning*.

The complexity of these three problems has been studied by different authors. Gottlob showed that these problems are either complete for Σ_2^{P} or Π_2^{P} [Got92]. Kautz and Selman [KS91] and Stillman [Sti90] proved NP- and coNP-completeness and polynomial time results for various syntactical restrictions of these problems.

We look at the above problems in the constraint satisfaction context, that means we restrict all formulas appearing in the input to be constraint formulas over a fixed Boolean constraint language Γ. Using the tools developed in Chapter 3 we achieve a full complexity classification for all three questions.

5.1 Reiter's Default Logic

A *default* d is an expression of the form

$$\frac{\alpha : \mathsf{M}\beta_1, \ldots, \mathsf{M}\beta_m}{\gamma},$$

where $\alpha, \beta_1, \ldots, \beta_m, \gamma$ are propositional formulas. We call α the *prerequisite*, $\beta_1, \ldots, \beta_m$ the *justification*, and γ the *consequence* of d. Intuitively d is a rule saying that if α is known and $\beta_1, \ldots, \beta_m$ are consistent with what we know, then γ is believed to be true. The rule mentioned in the example in the beginning of the chapter can be expressed by a default of the form

$$\frac{x \text{ is bird} : \mathsf{M}x \text{ can fly}}{x \text{ can fly}}.$$

A *default theory* T is a pair (W, D), where W is a set of propositional formulas and D a set of defaults. We refer to W as the *knowledge base* of T.

The set of formulas that can be derived to be believed from a default theory is formalized in the concept of extensions, which we define now. For a set S of propositional formulas we define $\mathrm{Th}(S)$ to be the deductive closure of S, i.e., the set of all formulas that are propositionally implied by S.

Definition 5.1. Let $T = (W, D)$ be a default theory. For a set S of propositional formulas let $\Delta(S)$ be the smallest set satisfying the following properties:

1. $W \subseteq \Delta(S)$
2. $\mathrm{Th}(\Delta(S)) = \Delta(S)$
3. If

$$\frac{\alpha : \mathsf{M}\beta_1, \ldots, \mathsf{M}\beta_m}{\gamma} \in D, \quad \alpha \in \Delta(S), \quad \text{and } \neg\beta_1, \ldots, \neg\beta_m \notin S,$$

then $\gamma \in \Delta(S)$.

A set of propositional formulas E is an *extension* for T, if E is a fixpoint of Δ, i.e., if $\Delta(E) = E$.

Note that if S is inconsistent and deductively closed, it holds that $\Delta(S) = \mathrm{Th}(W)$. Therefore T has an inconsistent extension if and only if the knowledge base W is inconsistent. In this case $\mathrm{Th}(W)$ is the only extension of T.

The following theorem provides an alternative definition of extensions.

Theorem 5.2 ([Rei80]). *Let $T = (W, D)$ a default theory and E a set of propositional formulas. Let $E_0 = W$ and*

$$E_{i+1} = \mathrm{Th}(E_i) \cup \left\{ \gamma \mid \frac{\alpha : \mathsf{M}\beta_1, \ldots, \mathsf{M}\beta_m}{\gamma} \in D,\ \alpha \in E_i \text{ and } \neg\beta_1, \ldots, \neg\beta_m \notin E \right\}$$

for all $i \in \mathbb{N}$. Then it holds that E is an extension for T if and only if $E = \bigcup_{i \in \mathbb{N}} E_i$.

The previous theorem shows that E is the deductive closure of W and the consequences added in the recursion. The next corollary formalizes this statement. For a default theory $T = (W, D)$ with extension E the set

$$\mathrm{gd}(E, T) =_{def} \left\{ \frac{\alpha : \mathsf{M}\beta_1, \ldots, \mathsf{M}\beta_m}{\gamma} \in D \mid \alpha \in E, \neg\beta_1, \ldots, \neg\beta_m \notin E \right\}$$

is called the set of *generating defaults* of E. For a set of defaults D, we denote the set of all consequences appearing in D by $\mathrm{cons}(D)$.

Corollary 5.3 ([Rei80]). *Let $T = (W, D)$ be a default theory and let E be an extension for T. Then the following equation is true.*

$$E = \mathrm{Th}(W \cup \mathrm{cons}(\mathrm{gd}(E, T)))$$

That means every extension is uniquely characterized by its generating defaults. So for finite default theories, i.e., default theories with a finite set of defaults and a finite knowledge base, all extensions have a finite representation. The size of this representation is bounded by the size of the given default theory T, because the generating defaults of an extension are a subset of the defaults in T.

The following three algorithmic questions have been investigated:

1. Given a default theory, does it have an extension? The existence of an extension means that the rules formulated in the defaults of the theory do not conflict in view of the underlying knowledge base.

2. Given a default theory T and a formula φ, does φ belong to *some* extension for T? This question is referred to as *credulous* or *brave reasoning*. The underlying intuition is that it is possible to conclude φ from T by applying defaults from T.

3. Given a default theory T and a formula φ, does φ belong to *every* extension for T? We speak of *skeptical* or *cautious* reasoning in this case. If φ is in every extension of T, it means that the defaults from T force us to conclude φ.

Georg Gottlob proved that the first two questions are Σ_2^{P}-complete problems and the third question is a Π_2^{P}-complete problem [Got92].

We take a closer look at these problems and investigate their complexities if we allow constraint formulas over some finite Boolean constraint language only. Let Γ be a finite Boolean constraint language. A Γ-default is an expression of the form

$$\frac{\alpha : \mathsf{M}\beta_1, \ldots, \mathsf{M}\beta_m}{\gamma},$$

where $\alpha, \beta_1, \ldots, \beta_m, \gamma$ are Γ-formulas. A Γ-default theory is a tuple $T = (W, D)$ where the knowledge base W is a set of Γ- formulas, and D is a set of Γ-defaults. If the prerequisite α is a tautology, e.g. the empty conjunction of constraint clauses, we write just

$$\frac{: \mathsf{M}\beta_1, \ldots, \mathsf{M}\beta_m}{\gamma}.$$

Since every constraint formula over a Boolean constraint language is equivalent to some propositional formula, we treat Γ-formulas as propositional formulas and Γ-default theories as default theories.

Let E be an extension for some Γ-default theory T. Note that E contains not only Γ-formulas and that E can even contain formulas that are not equivalent to any Γ-formula. However, according to Corollary 5.3, E is uniquely identified by its set of generating defaults which is a set of Γ-defaults.

In this chapter we investigate the complexity of the following three problems for every finite Boolean constraint language Γ.

Problem:	EXT(Γ)
Input:	a Γ-default theory $T = (W, D)$
Question:	does T have an extension?

Problem:	CRED(Γ)
Input:	a Γ-default theory $T = (W, D)$ and a Γ-formula φ
Question:	does φ belong to some extension for T?

Problem:	SCEPT(Γ)
Input:	a Γ-default theory $T = (W, D)$ and a Γ-formula φ
Question:	does φ belong to every extension for T?

The general upper complexity bounds for these problems follow directly from Gottlob's complexity results for unrestricted propositional default logic.

Proposition 5.4 ([Got92])**.** *Let Γ be a finite constraint language over $\{0, 1\}$. Then* EXT(Γ) $\in \Sigma_2^{\mathrm{P}}$, CRED($\Gamma$) $\in \Sigma_2^{\mathrm{P}}$, *and* SCEPT($\Gamma$) $\in \Pi_2^{\mathrm{P}}$.

First we show, that for two constraint languages that generate the same weak system without equality, each of the three problems above have the same complexity.

Proposition 5.5. *Let* Γ_1 *and* Γ_2 *be finite constraint languages over* $\{0,1\}$ *such that* $\Gamma_1 \subseteq \langle\Gamma_2\rangle_{\nexists,\neq}$. *Then* $\mathrm{EXT}(\Gamma_1) \leq_m^{log} \mathrm{EXT}(\Gamma_2)$, $\mathrm{CRED}(\Gamma_1) \leq_m^{log} \mathrm{CRED}(\Gamma_2)$, *and* $\mathrm{SCEPT}(\Gamma_1) \leq_m^{log} \mathrm{SCEPT}(\Gamma_2)$.

Proof. Let $T = (W, D)$ be a Γ_1-default theory and φ a Γ_1-formula. Due to Proposition 2.5, for every Γ_1-formula we can construct an equivalent Γ_2-formula in logarithmic space. Let $T' = (W', D')$ be the Γ_2-default theory that results from T by replacing every occurring Γ_1-formula by an equivalent Γ_2-formula and let φ' be a Γ_2-formula that is equivalent to φ. Then T and T' differ only in equivalent replacements, therefore they have exactly the same extensions and since φ and φ' are equivalent, φ' is in some or respectively all of these extensions if and only if φ is. This proves the three reductions stated in the proposition. □

In the end of this chapter we will see that the above proposition even holds if we only stipulate that $\Gamma_1 \subseteq \langle\Gamma_2\rangle$ instead of $\Gamma_1 \subseteq \langle\Gamma_2\rangle_{\nexists,\neq}$. However, there is no obvious way to generalize the above proof to give this stronger version of Proposition 5.5. That the complexity for all three questions depends only on the co-clone generated by the according constraint language will be a consequence of our classification.

5.2 Existence of an Extension

We first classify the complexity of $\mathrm{EXT}(\Gamma)$ depending on the constraint language Γ. The classifications for the constraint problems in the context of credulous and skeptical reasoning will be addressed afterwards.

With the first proposition we cover all cases, in which all formulas are 0-valid or all formulas are 1-valid. A formula is *0-valid* (*1-valid*) if it can be satisfied by setting all variables to 0 (to 1). We show that in these cases there exists a unique extension for every default theory. Considering that sets of 0-valid formulas are allways consistent as well as sets of 1-valid formulas, this result is not surprising. The proof relies on Reiter's characterization of extensions given in Theorem 5.2, which specifies a unique extension for every default theory over a constraint language mentioned in the proposition.

Proposition 5.6. *Let* Γ *be a finite constraint language over* $\{0,1\}$ *such that* $\langle\Gamma\rangle \subseteq \mathrm{II}_0$ *or* $\langle\Gamma\rangle \subseteq \mathrm{II}_1$. *Then every* Γ*-default theory has exactly one extension.*

Proof. Let $\langle\Gamma\rangle \subseteq \mathrm{II}_0$. Then every Γ-formula φ is 0-valid, i.e., the assignment that maps every variable from φ to 0 is a solution for φ. Let $T = (W, D)$ be a Γ-default theory. Then, according to Corollary 5.3, it holds that every extension is the deductive closure of a set of Γ-formulas. Obviously the deductive closure of 0-valid formulas contains only 0-valid formulas, therefore as well every extension of T contains only 0-valid formulas.

Now consider the construction of the set E_i in Theorem 5.2. Since every justification β of every Γ-default from D is 0-valid it follows that $\neg\beta$ is not 0-valid and therefore cannot appear in any extension of T. That means E_i does not depend on the choice of the extension E for every $i \in \mathbb{N}$. Since the union $\bigcup_{i\in\mathbb{N}} E_i$ again contains only 0-valid formulas, it is the unique extension of T.

For $\langle\Gamma\rangle \subseteq \mathrm{II}_1$ all Γ-formulas are 1-valid and the proof works analogously to the above. □

It turns out that these are the only easy cases in the sense that for all other Boolean constraint languages Γ than mentioned in the proposition, we will show later that the problem $\mathrm{EXT}(\Gamma)$ is hard for NP or even for Σ_2^{P}.

We now prove that for Schaefer constraint languages our problem is included in NP.

Proposition 5.7. *Let Γ be a finite constraint language over $\{0,1\}$ such that $\langle\Gamma\rangle \subseteq \mathrm{IE}_2$, $\langle\Gamma\rangle \subseteq \mathrm{IV}_2$, $\langle\Gamma\rangle \subseteq \mathrm{ID}_2$, or $\langle\Gamma\rangle \subseteq \mathrm{IL}_2$. Then $\mathrm{EXT}(\Gamma) \in \mathrm{NP}$.*

Proof. Note that a constraint language Γ satisfying the prerequisites of the proposition is Schaefer.

The following non-deterministic algorithm checks if a default theory has an extension by first guessing a set of defaults D' and then verifying that D' is a set of generating defaults of an extension:

1: **input:** a Γ-default theory $T = (W, D)$
2: guess a set $D' \subseteq D$ of generating defaults
3: let $\psi := \bigwedge_{\varphi\in W\cup\mathrm{cons}(D')} \varphi$
4: **for** every default $\frac{\alpha:\mathsf{M}\beta_1,\ldots,\mathsf{M}\beta_m}{\gamma} \in D'$ **do**
5: verify that $\psi \models \alpha$ and $\psi \not\models \neg\beta_i$ for every $1 \leq i \leq m$
6: **end for**
7: **for** every default $\frac{\alpha:\mathsf{M}\beta_1,\ldots,\mathsf{M}\beta_m}{\gamma} \in D \setminus D'$ **do**
8: verify that $\psi \not\models \alpha$ or $\psi \models \neg\beta_i$ for some $1 \leq i \leq m$
9: **end for**

We first show that the algorithm verifies that the set D' guessed in line 2 is a set of generating defaults of an extension for T, i.e., we show that $E =_{def} \mathrm{Th}(W \cup \mathrm{cons}(D'))$ is an extension for T with generating defaults D'.

Note that the formula ψ defined in line 3 is a conjunction of Γ-formulas and therefore itself is a Γ-formula. It also holds $E = \mathrm{Th}(\{\psi\})$ and it holds $\varphi \in E$ if and only if $\psi \models \varphi$ for every propositional formula φ.

In lines 4-6 it is verified that all defaults $\frac{\alpha:\mathsf{M}\beta_1,\ldots,\mathsf{M}\beta_m}{\gamma} \in D'$ satisfy the conditions $\alpha \in E$ and $\neg\beta_1, \ldots, \neg\beta_m \notin E$, which ensure that $\gamma \in \Delta(E)$. Therefore $\mathrm{cons}(D') \subseteq \Delta(E)$ and since $W \subseteq \Delta(E)$ and $\Delta(E)$ is deductively closed, it follows $\Delta(E) \subseteq E$.

Lines 7-9 check that for all defaults $\frac{\alpha:\mathsf{M}\beta_1,\ldots,\mathsf{M}\beta_m}{\gamma} \in D \setminus D'$ at least one of the conditions $\alpha \in E$ and $\neg\beta_1, \ldots, \neg\beta_m \notin E$ is not satisfied. That means these defaults do not affect $\Delta(E)$. So, because of the minimality of $\Delta(E)$, it holds that $E = \Delta(E)$, which means E is an extension for T with generating defaults D'.

We now have a closer look on the running time of the algorithm. Note that $\psi \models \varphi$ is true if and only if ψ is equivalent to $\psi \wedge \varphi$ and $\psi \models \neg\varphi$ is true if and only if $\psi \wedge \varphi$ is not satisfiable. Since Γ is Schaefer, we can decide in polynomial time whether two Γ-formulas are equivalent according to [BHRV02] and whether a Γ-formula is satisfiable in polynomial time according to [Sch78]. Hence, lines 5 and 8 can be implemented to perform in polynomial time, and so the above algorithm is an non-deterministic polynomial time algorithm.

□

To complete the complexity classification for $\mathrm{EXT}(\Gamma)$, we identify the NP- and Σ_2^{P}-hard cases. The next lemma provides implementation results that are helpful for our hardness results. In the proof we work with irredundant weak bases that are introduced in Chapter 3.

Lemma 5.8. *Let Γ be a finite constraint language over $\{0,1\}$. The following is true:*

1. *if $\mathrm{IR}_2 \subseteq \langle\Gamma\rangle$, then $\mathrm{C}_0 \times \mathrm{C}_1 \in \langle\Gamma\rangle_{\not\equiv,\neq}$*
2. *if $\langle\Gamma\rangle \subseteq \mathrm{IN}_2$ and $\langle\Gamma\rangle \not\subseteq \mathrm{IN}$, then $\mathrm{Odd}^2 \in \langle\Gamma\rangle_{\not\equiv,\neq}$.*

Proof. 1. Let $\mathrm{IR}_2 \subseteq \langle\Gamma\rangle$. Let s be a core-size of $\langle\Gamma\rangle$. Due to Theorem 3.9 it holds that

$$R =_{def} \mathrm{Pol}(\Gamma)(\mathrm{Cols}_s)$$

is a weak base for $\langle\Gamma\rangle$. Note that according to Table 3.2 we have $s \geq 1$. Let

$$\varphi =_{def} R(\underbrace{x,\ldots,x}_{2^{s-1}},\underbrace{y,\ldots,y}_{2^{s-1}}).$$

We show that $\mathrm{C}_0 \times \mathrm{C}_1(x,y) \equiv \varphi$. Since $\mathrm{C}_0 \times \mathrm{C}_1 = \{(0,1)\}$, we have to show that the assignment $I : \{x,y\} \to \{0,1\}$ with $I(x) =_{def} 0$ and

$I(y) =_{def} 1$ is the only solution for φ. Since $\mathrm{Pol}(\Gamma) \subseteq \mathrm{R}_2$, it holds that every polymorphism is both 0-reproducing and 1-reproducing. Consider Cols_s. Its leftmost column contains only 0s and its rightmost column contains only 1s. So if we apply $\mathrm{Pol}(\Gamma)$ these columns still contain only 0s or only 1s. Hence, $R(-,1) = (0,\ldots,0)$ and $R(-,2^s) = (1,\ldots,1)$ and every solution for φ maps x to 0 and y to 1.

It is easy to see, that I indeed is a solution of φ, because

$$\mathrm{Cols}_s(1,-) = (\underbrace{0,\ldots,0}_{2^{s-1}},\underbrace{1,\ldots,1}_{2^{s-1}}).$$

Thus, $\mathrm{C}_0 \times \mathrm{C}_1(x,y) \equiv \varphi$ which means $\mathrm{C}_0 \times \mathrm{C}_1 \in \langle R\rangle_{\not\exists} \subseteq \langle\Gamma\rangle_{\not\exists}$. Since $\mathrm{C}_0 \times \mathrm{C}_1$ is irredundant, it follows $\mathrm{C}_0 \times \mathrm{C}_1 \in \langle\Gamma\rangle_{\not\exists,\neq}$ due to Proposition 3.11.

2. Let $\langle\Gamma\rangle \subseteq \mathrm{IN}_2$ and $\langle\Gamma\rangle \not\subseteq \mathrm{IN}$. Let s be a core-size of $\langle\Gamma\rangle$ and let

$$R =_{def} \mathrm{Pol}(\Gamma)(\mathrm{Cols}_s).$$

Define S to be the 2-ary Boolean relation such that

$$S(x,y) \equiv R(\underbrace{x,\ldots,x}_{2^{s-1}},\underbrace{y,\ldots,y}_{2^{s-1}}).$$

We show that $S = \mathrm{Odd}^2$. Note that $\neg \in \mathrm{Pol}(\Gamma)$ because $\langle\Gamma\rangle \subseteq \mathrm{IN}_2$. It holds that

$$\mathrm{Cols}_s(1,-) = (\underbrace{0,\ldots,0}_{2^{s-1}},\underbrace{1,\ldots,1}_{2^{s-1}}).$$

Since $\mathrm{Cols}_s \subseteq R$ and $\neg \in \mathrm{Pol}(\Gamma)$ is a polymorphism of R, it follows that $\{(0,1),(1,0)\} = \mathrm{Odd}^2 \subseteq S$. Now assume $(0,0) \in S$, which means $(0,\ldots,0) \in R$. This implies that c_0 is a polymorphism of R. Since $\neg$ is a polymorphism of R as well, it then follows $\langle\Gamma\rangle \subseteq \mathrm{IN}$, which contradicts the given prerequisites. So it holds $(0,0) \notin S$ and analogously it holds $(1,1) \notin S$. Thus, $S = \mathrm{Odd}^2$.

Therefore we have $\mathrm{Odd}^2 \in \langle R\rangle_{\not\exists} \subseteq \langle\Gamma\rangle_{\not\exists}$, and since Odd^2 is irredundant, it follows $\mathrm{Odd}^2 \in \langle\Gamma\rangle_{\not\exists,\neq}$. □

We now prove that $\mathrm{EXT}(\Gamma)$ is NP-hard if Γ generates a co-clone above IR_2. With the previous lemma the proposition easily follows from an NP-hardness result from Kautz and Selman for default logics that are build only from literals [KS91].

Proposition 5.9. *Let Γ be a finite constraint language over $\{0,1\}$ such that $\mathrm{IR}_2 \subseteq \langle\Gamma\rangle$. Then $\mathrm{EXT}(\Gamma)$ is NP-hard.*

Proof. Due to Lemma 5.8 it holds that $\mathrm{C}_0 \times \mathrm{C}_1 \in \langle\Gamma\rangle_{\not\exists,\neq}$. Since it follows from Proposition 5.5 that $\mathrm{EXT}(\mathrm{C}_0 \times \mathrm{C}_1) \leq_m^{\log} \mathrm{EXT}(\Gamma)$, it is sufficient to show that $\mathrm{EXT}(\mathrm{C}_0 \times \mathrm{C}_1)$ is NP-hard.

We use the following result from Kautz and Selman: It is NP-hard to decide if a given default theory in which all formulas are conjunctions of literals has an extension [KS91]. Note that $\top$ and $\bot$ appearing in their notation can be expressed by the empty conjunction of literals and by conjunction of contradicting literals. Translated to our notation this result can be formulated as $\mathrm{EXT}(\{\mathrm{C}_0, \mathrm{C}_1\})$ is NP-hard. Note that the result from Kautz and Selman was proven for a more restricted type of default theories, where the defaults are required to match a special form.

Let $T = (W, D)$ be a default theory over $\{\mathrm{C}_0, \mathrm{C}_1\}$ and let f and t be variables not appearing in T. To construct a $\{\mathrm{C}_0 \times \mathrm{C}_1\}$-default theory exchange every clause $\mathrm{C}_0(x)$ in T by $\mathrm{C}_0 \times \mathrm{C}_1(x, t)$ and every clause $\mathrm{C}_1(x)$ in T by $\mathrm{C}_0 \times \mathrm{C}_1(f, x)$. Additionally add the clause $\mathrm{C}_0 \times \mathrm{C}_1(f, t)$ to the knowledge base W. It is easy to see, that the old default theory has an extension if and only if the new one has one. So this reduction transfers the NP-hardness from Kautz and Selman to $\mathrm{EXT}(\mathrm{C}_0 \times \mathrm{C}_1)$, which completes the proof. □

Now we show NP-hardness for the cases IN_2, IL_3 and ID. However, the result for IN_2 is not optimal as we will see in Proposition 5.11.

Proposition 5.10. *Let Γ be a finite constraint language over $\{0, 1\}$ such that $\mathrm{ID} \subseteq \langle\Gamma\rangle \subseteq \mathrm{IN}_2$. Then $\mathrm{EXT}(\Gamma)$ is NP-hard.*

Proof. Due to Lemma 5.8 it holds $\mathrm{Odd}^2 \in \langle\Gamma\rangle_{\not\exists,\neq}$, therefore it follows with Proposition 5.5 that $\mathrm{EXT}(\mathrm{Odd}^2) \leq_m^{\log} \mathrm{EXT}(\Gamma)$. To complete the proof we show that $\mathrm{EXT}(\mathrm{Odd}^2)$ is NP-hard.

Similarly as in the proof of Proposition 5.9, the NP-hardness of $\mathrm{EXT}(\mathrm{Odd}^2)$ follows from the NP-hardness of $\mathrm{EXT}(\{\mathrm{C}_0, \mathrm{C}_1\})$ which is proven in [KS91].

Let $T = (W, D)$ be a $\{\mathrm{C}_0, \mathrm{C}_1\}$-default theory. Let t and f be new variables and for a $\{\mathrm{C}_0, \mathrm{C}_1\}$-formula φ let φ' be the formula obtained from φ by replacing every clause $\mathrm{C}_0(x)$ by $\mathrm{Odd}^2(x, t)$ and every clause $\mathrm{C}_1(x)$ by $\mathrm{Odd}^2(x, f)$. Let T' be the default theory that is obtained by exchanging every formula φ in T by φ' and adding the clause $\mathrm{Odd}^2(f, t)$ to the knowledge base.

For $\{\mathrm{C}_0, \mathrm{C}_1\}$-formulas φ and ψ it holds that $\varphi \vDash \psi$ if and only if $\varphi' \wedge \mathrm{Odd}^2(t, f) \vDash \psi'$ and it holds that $\varphi \vDash \neg\psi$ if and only if $\varphi' \wedge \mathrm{Odd}^2(t, f) \vDash \neg\psi'$. That means a set $F \subseteq D$ is a set of generating defaults of an extension for T if and only if the corresponding set F' where every formula φ from F is replaced by φ' is a set of generating defaults of an extension for T'.

Since T' can be constructed in logarithmic space, it follows

$$\mathrm{EXT}(\{\mathrm{C}_0, \mathrm{C}_1\}) \leq_m^{\log} \mathrm{EXT}(\mathrm{Odd}^2).$$

□

The following two propositions prove Σ_2^{P}-hardness for the cases IN_2 and II_2. The proofs are very similar and direct modifications of Gottlob's hardness proof for the question if some (non-restricted) default theory has an extension. In both cases we reduce from the complement of $\mathrm{QBF}_2(\Gamma)$. The Problem $\mathrm{QBF}_2(\Gamma)$ is the question if a given quantified formula of the type $\forall x_1, \ldots, x_k \exists y_1, \ldots, y_l \varphi$, where φ is a constraint formula over Γ with $\mathrm{Var}(\varphi) = \{x_1, \ldots, x_k, y_1, \ldots, y_l\}$, is valid. Hemaspaandra proved that $\mathrm{QBF}_2(\Gamma)$ is complete for Π_2^{P}, if Γ is not Schaefer [Hem04].

Proposition 5.11. *Let Γ be a finite constraint language over $\{0,1\}$ such that $\langle\Gamma\rangle = \mathrm{IN}_2$. Then $\mathrm{EXT}(\Gamma)$ is Σ_2^{P}-hard.*

Proof. Since $\langle\Gamma\rangle \not\subseteq \mathrm{IN}$ (see Figure 2.3), we know from Lemma 5.8 that $\mathrm{Odd}^2 \in \langle\Gamma\rangle_{\not\equiv,\neq}$. Therefore it holds $\left\langle \Gamma \cup \left\{\mathrm{Odd}^2\right\}\right\rangle_{\not\equiv,\neq} = \langle\Gamma\rangle_{\not\equiv,\neq}$ and it follows $\mathrm{EXT}(\Gamma \cup \left\{\mathrm{Odd}^2\right\}) \leq_m^{\log} \mathrm{EXT}(\Gamma)$ due to Proposition 5.5. So it is sufficient to prove Σ_2^{P}-hardness of $\mathrm{EXT}(\Gamma \cup \left\{\mathrm{Odd}^2\right\})$.

We show $\overline{\mathrm{QBF}_2(\Gamma)} \leq_m^{\log} \mathrm{EXT}(\Gamma \cup \left\{\mathrm{Odd}^2\right\})$. Let

$$\psi =_{def} \forall x_1, \ldots, x_k \exists y_1, \ldots y_l \varphi$$

be an instance for $\mathrm{QBF}_2(\Gamma)$, i.e., φ is a Γ-formula and it holds that $\mathrm{Var}(\varphi) = \{x_1, \ldots, x_k, y_1, \ldots, y_l\}$. Let $x'_1, \ldots, x'_k, z$ be distinct variables not appearing in φ. We construct a default theory $T = (W, D)$ over $\Gamma \cup \left\{\mathrm{Odd}^2\right\}$ by setting:

$$W =_{def} \{\mathrm{Odd}(x_i, x'_i) \mid 1 \leq i \leq k\}$$

and

$$\begin{aligned} D =_{def} & \left\{ \frac{: \mathsf{M}\mathrm{Odd}^2(x_i, x_{i+1})}{\mathrm{Odd}^2(x_i, x_{i+1})} \mid 1 \leq i \leq k-1 \right\} \\ & \cup \left\{ \frac{: \mathsf{M}\mathrm{Odd}^2(x_i, x'_{i+1})}{\mathrm{Odd}^2(x_i, x'_{i+1})} \mid 1 \leq i \leq k-1 \right\} \\ & \cup \left\{ \frac{: \mathsf{M}\varphi}{\mathrm{Odd}^2(z, z)} \right\} \end{aligned}$$

The idea behind this construction is the following: we use the the variables x'_i to express $\neg x_i$. The defaults in the first two sets simulate assignments to the $\forall$-quantified variables $x_1, \ldots, x_k$ by forcing each possible extension to contain for any two variables x_i and x_j either $x_i = x_j$ or $x_i \neq x_j$. The last default guarantees that $\neg\varphi$ is valid in each extension, because it has a non-consistent

consequence. Note that this construction is similar to the construction from Gottlob in the Σ_2^{P}-hardness proof for the question if a default theory has an extension [Got92].

We prove that T has an extension if and only if ψ is not valid. First assume that T has an extension E. Since W is consistent, E is consistent as well. Consider the default

$$\frac{:\mathsf{M}\varphi}{\mathrm{Odd}^2(z,z)} \in D.$$

If $\neg\varphi \notin E$, then the default implies $\mathrm{Odd}^2(z,z) \in E$, which contradicts the consistency of E. This means $\neg\varphi \in E$. Since the $\exists$-quantified variables $y_1, \ldots, y_l$ do not appear in T otherwise than in the justification of the considered default, it holds that satisfying assignments for all formulas in E do not depend on the variables $y_1, \ldots, y_l$. It follows that there exists an assignment to the variables $x_1, \ldots, x_k$ such that every extension to $\mathrm{Var}(\varphi)$ satisfies $\neg\varphi$. Thus, ψ is not valid.

For the other direction let ψ be not valid. Then there is an assignment I to $x_1, \ldots, x_k$, such that every extension of I to $\mathrm{Var}(\varphi)$ satisfies $\neg\varphi$. We construct an extension for T. Let

$$S_0 =_{def} \left\{\mathrm{Odd}^2(x_i, x'_{i+1}) \mid 1 \leq i \leq k-1 \text{ and } I(x_i) = I(x_{i+1})\right\}$$

and

$$S_1 =_{def} \left\{\mathrm{Odd}^2(x_i, x_{i+1}) \mid 1 \leq i \leq k-1 \text{ and } I(x_i) \neq I(x_{i+1})\right\}.$$

We prove that $E =_{def} \mathrm{Th}(W \cup S_0 \cup S_1)$ is an extension for T, i.e., we prove that $\Delta(E) = E$. First note that E is consistent. Since for every $\psi \in S_0 \cup S_1$ there is a default $\frac{:\mathsf{M}\psi}{\psi} \in D$, it follows $E \subseteq \Delta(E)$.

Assume $E \subsetneq \Delta(E)$. Then there is some default $d \in D$ such that its consequence γ is not in E and its justification is consistent with E, which implies $\gamma \in \Delta(E)$. If d is of the form $\frac{:\mathsf{M}\mathrm{Odd}^2(x_i,x_{i+1})}{\mathrm{Odd}^2(x_i,x_{i+1})}$ for some $i \in \{1, \ldots, k_1\}$, then $\mathrm{Odd}^2(x_i, x'_{i+1}) \in E$. Since $\mathrm{Odd}^2(x_{i+1}, x'_{i+1}) \in W \subseteq E$, it holds that the justification $\beta = \mathrm{Odd}^2(x_i, x_{i+1})$ is not consistent to E in this case. Analogously the justification is not consistent to E, if d is of the form $\frac{:\mathsf{M}\mathrm{Odd}^2(x_i,x'_{i+1})}{\mathrm{Odd}^2(x_i,x'_{i+1})}$.

So, d is the default $\frac{:\mathsf{M}\varphi}{\mathrm{Odd}^2(z,z)}$. Let J be an assignment that satisfies every formula from $W \cup S_0 \cup S_1$. Obviously it holds either that $J|_{\{x_1,\ldots,x_k\}} = I$, or that $J|_{\{x_1,\ldots,x_k\}} = I'$ which is defined by $I'(x) = \neg I(x)$. Recall that every extension of I to $\mathrm{Var}(\varphi)$ satisfies $\neg\varphi$. Since $\Gamma \subseteq \mathrm{IN}_2$, it holds that φ is complementive therefore $\neg\varphi$ is complementive as well and every extension of I' to $\mathrm{Var}(\varphi)$ satisfies $\neg\varphi$ as well. Hence, $\neg\varphi \in E$, so the justification of D is not consistent to E.

Thus we showed that $E = \Delta(E)$, i.e that E is an extension of T. It is easy to see that T can be constructed in logarithmic space, so we proved $\overline{\mathrm{QBF}_2(\Gamma)} \leq_m^{\log} \mathrm{EXT}(\Gamma \cup \{\mathrm{Odd}^2\})$. Since $\langle\Gamma\rangle = \mathrm{IN}_2$, it holds that $\mathrm{QBF}_2(\Gamma)$ is Π_2^{P}-complete due to [Hem04], which completes the proof. □

To complete the classification we cover the case of the largest Boolean co-clone II_2 in the next proposition. The proof is only a slight modification from Gottlob's proof for Σ_2^{P}-hardness of the problem whether a default theory has an extension.

Proposition 5.12. *Let Γ be a finite constraint language over $\{0,1\}$ such that $\langle\Gamma\rangle = \mathrm{II}_2$. Then $\mathrm{EXT}(\Gamma)$ is Σ_2^{P}-complete.*

Proof. This proof is similar to the proof of Proposition 5.11. Since according to Lemma 5.8 it holds $\mathrm{C}_0, \mathrm{C}_1 \in \langle\Gamma\rangle_{\not\equiv,\neq}$, and since $\mathrm{QBF}_2(\Gamma)$ is Π_2^{P}-complete for $\langle\Gamma\rangle = \mathrm{II}_2$ [Hem04], it is enough to show $\overline{\mathrm{QBF}_2(\Gamma)} \leq_m^{\log} \mathrm{EXT}(\Gamma \cup \{\mathrm{C}_0 \times \mathrm{C}_1\})$. Let

$$\psi =_{def} \forall x_1, \ldots, x_k \exists y_1, \ldots y_l \varphi$$

be an instance for $\mathrm{QBF}_2(\Gamma)$, i.e., φ is a constraint formula over Γ such that $\mathrm{Var}(\varphi) = \{x_1, \ldots, x_k, y_1, \ldots, y_l\}$. Let f and t be two different variables not appearing in φ. We construct a $\Gamma \cup \{\mathrm{C}_0 \times \mathrm{C}_1\}$-default $T = (W, D)$ by defining:

$$W =_{def} \{\mathrm{C}_0 \times \mathrm{C}_1(f,t)\},$$

$$\begin{aligned} D =_{def} & \left\{ \frac{: \mathsf{M}\mathrm{C}_0 \times \mathrm{C}_1(x_i,t)}{\mathrm{C}_0 \times \mathrm{C}_1(x_i,t)} \mid 1 \leq i \leq k-1 \right\} \\ & \cup \left\{ \frac{: \mathsf{M}\mathrm{C}_0 \times \mathrm{C}_1(f,x_i)}{\mathrm{C}_0 \times \mathrm{C}_1(f,x_i)} \mid 1 \leq i \leq k \right\} \\ & \cup \left\{ \frac{: \mathsf{M}\varphi}{\mathrm{C}_0 \times \mathrm{C}_1(t,t)} \right\} \end{aligned}$$

The variables t and f simulate the constants 0 and 1. The defaults in the first two sets simulate assignments for the variables $x_1, \ldots, x_k$ and the last default makes sure that every extension contains $\neg\varphi$.

It holds that T has an extension if and only if ψ is not valid. We omit the proof for this because it works analogously to the proof of Proposition 5.11. Since T can be constructed in logarithmic space, this completes the proof. □

Now we summarize the results of this section in the following trichotomy theorem:

Theorem 5.13. *Let* Γ *be a finite contraint language over* $\{0,1\}$*. Then the following holds:*

- *If* $\mathrm{IN}_2 \subseteq \langle\Gamma\rangle$*, then* $\mathrm{EXT}(\Gamma)$ *is* Σ_2^{P}*-complete.*
- *Otherwise, if* $\mathrm{IR}_2 \subseteq \langle\Gamma\rangle$ *or* $\mathrm{ID} \subseteq \langle\Gamma\rangle$*, then* $\mathrm{EXT}(\Gamma)$ *is* NP*-complete.*
- *In all other cases* $\mathrm{EXT}(\Gamma)$ *is decidable in polynomial time.*

5.3 Credulous and Skeptical Reasoning

It is easy to see that credulous reasoning cannot be easier than the question if there exists an extension at all: since the knowledge base W of a default theory $T = (W, D)$ is a subset of every extension of T, it holds that T has an extension if and only T has an extension that contains $\bigwedge_{\varphi \in W} \varphi$. This proves the next proposition.

Proposition 5.14. *Let* Γ *be a finite constraint language over* $\{0,1\}$*. Then* $\mathrm{EXT}(\Gamma) \leq_m^{log} \mathrm{CRED}(\Gamma)$.

In Proposition 5.6 we showed that in the case of 0-valid or 1-valid constraint languages there always exists a unique extension. Therefore for such constraint languages credulous reasoning and skeptical reasoning are the same questions. We show that both problems are in P for constraint languages that additionally are Schaefer, and coNP-complete for constraint languages that additionally are non-Schaefer.

Proposition 5.15. *Let* Γ *be a finite constraint language over* $\{0,1\}$ *such that* $\langle\Gamma\rangle \subseteq \mathrm{II}_0$ *or* $\langle\Gamma\rangle \subseteq \mathrm{II}_1$*. Then the following holds:*

- *If* $\mathrm{IN} \subseteq \langle\Gamma\rangle$*, then* $\mathrm{CRED}(\Gamma)$ *and* $\mathrm{SCEPT}(\Gamma)$ *are* coNP*-complete.*
- *Otherwise* $\mathrm{CRED}(\Gamma)$ *and* $\mathrm{SCEPT}(\Gamma)$ *are in* P*.*

Proof. In this proof we assume $\langle\Gamma\rangle \subseteq \mathrm{II}_0$, for $\langle\Gamma\rangle \subseteq \mathrm{II}_1$ the result follows analogously.

Since $\langle\Gamma\rangle \subseteq \mathrm{II}_0$ every Γ-default theory T has a unique extension $E(T)$ due to Proposition 5.6. So, a formula is in some extension of T if and only if it is in every extension of T, which means $\mathrm{CRED}(\Gamma) = \mathrm{SCEPT}(\Gamma)$.

There are two cases to consider:

Case 1: $\mathrm{IN} \not\subseteq \langle\Gamma\rangle$. In this case it holds $\langle\Gamma\rangle \subseteq \mathrm{IV}_0$, $\langle\Gamma\rangle \subseteq \mathrm{IE}_0$, $\langle\Gamma\rangle \subseteq \mathrm{IL}_0$, or $\langle\Gamma\rangle \subseteq \mathrm{ID}_0$, i.e., Γ is Schaefer.

We give a polynomial time algorithm that, given a Γ-default theory T and a Γ formula φ, constructs the extension of T and checks whether φ belongs to it. The algorithm accepts if this is *not* the case, and rejects otherwise. The reason why we give an algorithm that decides $\overline{\text{CRED}(\Gamma)}$ rather than CRED(Γ) is that we want to use it in the next case to prove a coNP-result.

1: **input** a Γ-default theory $T = (W, D)$, a Γ-formula φ
2: let $E := W$
3: let $E' := \emptyset$
4: **while** $E \neq E'$ **do**
5: $E' := E$
6: **for** all $\frac{\alpha:\mathsf{M}\beta_1,\ldots,\mathsf{M}\beta_m}{\gamma} \in D$ **do**
7: **if** $E' \models \alpha$ **then**
8: $E := E \cup \{\gamma\}$
9: **end if**
10: **end for**
11: **end while**
12: **if** $E \models \varphi$ **then**
13: reject
14: **else**
15: accept
16: **end if**

Lines 2-11 implement the construction in Theorem 5.2. Since $E(T)$ contains only 0-valid formulas, for every justification β the condition that $\neg\beta \notin E(T)$ is true. So it is easy to see, that, if we execute lines 5-10, and if we have $\text{Th}(E) = \text{Th}(E_i)$ for the set E_i from Theorem 5.2 before the execution, it holds $\text{Th}(E) = \text{Th}(E_{i+1})$ after the execution. Since we start with $E = W$, it holds that $\text{Th}(E) = E(T)$ after executing lines 2-11.

Thus, the algorithm checks in line 12, whether $\varphi \in E(T)$ and rejects if this is the case and accepts otherwise.

Note that, for some set S of formulas and a formula ψ it holds that $S \models \psi$ if and only if the conjunction of all formulas from S implies ψ. Since Γ is Schaefer, the implication problem for Γ-formulas is solvable in polynomial time (this follows from [BHRV02] and can be found explicitly in [SS07b]). Hence, the above algorithm decides CRED(Γ) and SCEPT(Γ) in polynomial time.

Case 2: $\text{IN} \subseteq \langle\Gamma\rangle$. Note that in this case the implication problem for Γ-formulas

is complete for coNP [SS07b], therefore the above algorithm does not prove decidability in polynomial time for CRED(Γ) and SCEPT(Γ).

The coNP-hardness of CRED(Γ) and SCEPT(Γ) follows easily from the coNP-completeness of the implication problem for Γ-formulas, because if φ and ψ are Γ-formulas, then it holds $\varphi \models \psi$ if and only if ψ is in the unique extension of $T = (\{\varphi\}, \emptyset)$.

To show that CRED(Γ) and SCEPT(Γ) are in coNP we modify the above algorithm to get a non-deterministic algorithm that accepts if and only if $\varphi \notin E(T)$. We replace line 7 by:

7a: guess an assignment I that satisfies every formula in E'
7b: **if** I satisfies α **then**

And we replace line 12 by:

12a: guess an assignment I that satisfies every formula in E
12b: **if** I satisfies φ **then**

The guessed assignments are meant to be assignments for all variables appearing in T and φ. First note that we now have a non-deterministic algorithm that runs in polynomial time, since for testing if an assignment satisfies a Γ-formula we just have to test if the assignment satisfies every clause of the formula.

We now prove that the algorithm is correct. First we look at the new lines 7a and 7b. If $E' \models \alpha$, then every assignment I satisfying every formula in E, satisfies α as well, therefore γ is added to E in line 8. If $E' \not\models \alpha$, then there exists an assignment that satisfies every formula from E' but not α. If the algorithm guesses this assignment, then γ is not added to E in line 8, otherwise it is added. That means, if the algorithm guesses in each iteration an assignment that is a counterexample for $E' \models \alpha$ if there is one, then it adds exactly the same to E as the polynomial time algorithm without modification. So, the assignments I can be guessed such that $\mathrm{Th}(E) = E(T)$ after the execution of the while loop. If, for some non-deterministic choice, $E' \models \alpha$ does not hold and the algorithm does not guess a counter-example, then γ is added even if it might not be in the extension $E(T)$. In this case we have $E(T) \subseteq \mathrm{Th}(E)$.

Now assume $\varphi \notin E(T)$. If the algorithm guesses such that $\mathrm{Th}(E) = E(T)$ after the execution of the while loop, then there is a satisfying assignment satisfying all formulas from E, that satisfies not φ. Thus, guessing this in line 12a the algorithm accepts.

For the converse direction let $\varphi \in E(T)$. According to the above it holds, for every possible guess of I in every iteration of the while loop, that $E(T) \subseteq \mathrm{Th}(E)$ after the execution of the while loop. Therefore $\varphi \in \mathrm{Th}(E)$ and every possible assignment I the algorithm can choose in line 12a satisfies φ, which means the algorithm rejects.

Hence, the outlined algorithm decides $\overline{\mathrm{CRED}(\Gamma)}$ and $\overline{\mathrm{SCEPT}(\Gamma)}$ in nondeterministic polynomial time. Therefore $\mathrm{CRED}(\Gamma)$ and $\mathrm{SCEPT}(\Gamma)$ are in coNP. □

In the cases where we showed NP-completeness for $\mathrm{EXT}(\Gamma)$, we get NP- and coNP-completeness for $\mathrm{CRED}(\Gamma)$ and $\mathrm{SCEPT}(\Gamma)$ with similar proofs.

Proposition 5.16. *Let* Γ *be a finite constraint language over* $\{0,1\}$ *such that* Γ *is Schaefer and such that* $\mathrm{IR}_2 \subseteq \langle\Gamma\rangle$ *and* $\mathrm{ID} \subseteq \langle\Gamma\rangle$. *Then* $\mathrm{CRED}(\Gamma)$ *is* NP*-complete and* $\mathrm{SCEPT}(\Gamma)$ *is* coNP*-complete.*

Proof. To see that $\mathrm{CRED}(\Gamma) \in \mathrm{NP}$ consider the algorithm given in the Proof of Proposition 5.7. Give the algorithm a Γ-formula χ as additional input and add the following line in the end:

10: verify that $\psi \models \chi$

All verifications of the algorithm have a positive result if and only if the algorithm guesses the generating defaults of an extension that contains χ. Since Γ is Schaefer, the new verification can be done in polynomial time in the same way as the other verifications in the algorithm. This shows that $\mathrm{CRED}(\Gamma) \in \mathrm{NP}$.

Adding instead the line

10: verify that $\psi \not\models \chi$

gives an NP-algorithm for $\overline{\mathrm{SCEPT}(\Gamma)}$, which implies that $\mathrm{SCEPT}(\Gamma) \in \mathrm{coNP}$.

The NP-hardness of $\mathrm{CRED}(\Gamma)$-follows from Propositions 5.9, 5.10 and 5.14. For the coNP-hardness we use a result from Kautz and Selman which can be formulated as: $\mathrm{SCEPT}(\{\mathrm{C}_0, \mathrm{C}_1\})$ is coNP-hard [KS91]. Reductions very similar to those in the proofs for Propositions 5.9 and 5.10 transfer the coNP-hardness of $\mathrm{SCEPT}(\{\mathrm{C}_0, \mathrm{C}_1\})$ to $\mathrm{SCEPT}(\Gamma)$. □

Finally we show Σ_2^{P}- and Π_2^{P}-completeness results for constraint languages generating IN_2 or II_2. Again the arguments in the proof are very similar to those used for the corresponding cases in the previous section (see Propositions 5.11 and 5.12).

Proposition 5.17. *Let* Γ *be a finite constraint language over* $\{0,1\}$ *such that* $\mathrm{IN}_2 \subseteq \langle\Gamma\rangle$. *Then* $\mathrm{CRED}(\Gamma)$ *is* Σ_2^{P}*-complete and* $\mathrm{SCEPT}(\Gamma)$ *is* Π_2^{P}*-complete.*

Proof. The upper complexity bounds follow from Proposition 5.4. The Σ_2^{P}-hardness of CRED(Γ)-follows from Propositions 5.11, 5.12 and 5.14.

The Π_2^{P}-hardness of SCEPT(Γ) can be proven by a slight modification of Gottlob's proof for the coNP-hardness of skeptical reasoning. We make a reduction from $\mathrm{QBF}_2(\Gamma)$ similar to the reductions in Propositions 5.11 and 5.12. Let

$$\psi =_{def} \forall x_1, \ldots, x_k \exists y_1, \ldots y_l \varphi$$

be an instance for $\mathrm{QBF}_2(\Gamma)$, i.e., φ is a Γ-formula and it holds that $\mathrm{Var}(\varphi) = \{x_1, \ldots, x_k, y_1, \ldots, y_l\}$. Let $x'_1, \ldots, x'_k, t, f$ be distinct variables not appearing in φ.

If $\langle\Gamma\rangle = \mathrm{IN}_2$ we construct the default theory $T = (W, D)$ by setting:

$$W_1 =_{def} \{\mathrm{Odd}(x_i, x'_i) \mid 1 \leq i \leq k\}$$

and

$$\begin{aligned} D =_{def} & \left\{ \frac{: \mathsf{M}\mathrm{Odd}^2(x_i, x_{i+1})}{\mathrm{Odd}^2(x_i, x_{i+1})} \mid 1 \leq i \leq k-1 \right\} \\ & \cup \left\{ \frac{: \mathsf{M}\mathrm{Odd}^2(x_i, x'_{i+1})}{\mathrm{Odd}^2(x_i, x'_{i+1})} \mid 1 \leq i \leq k-1 \right\} \\ & \cup \left\{ \frac{: \mathsf{M}\varphi}{\varphi} \right\} \end{aligned}$$

For $\langle\Gamma\rangle = \mathrm{II}_2$ we construct the default theory $T = (W, D)$ by defining:

$$W =_{def} \{\mathrm{C}_0 \times \mathrm{C}_1(f, t)\},$$

$$\begin{aligned} D =_{def} & \left\{ \frac{: \mathsf{M}\mathrm{C}_0 \times \mathrm{C}_1(x_i, t)}{\mathrm{C}_0 \times \mathrm{C}_1(x_i, t)} \mid 1 \leq i \leq k \right\} \\ & \cup \left\{ \frac{: \mathsf{M}\mathrm{C}_0 \times \mathrm{C}_1(f, x_i)}{\mathrm{C}_0 \times \mathrm{C}_1(f, x_i)} \mid 1 \leq i \leq k \right\} \\ & \cup \left\{ \frac{: \mathsf{M}\varphi}{\varphi} \right\} \end{aligned}$$

It holds that ψ is valid if and only if φ belongs to each extension for T. The proof uses similar arguments as the proof for Proposition 5.11, therefore we omit it here.

Since for $\langle\Gamma\rangle = \mathrm{IN}_2$ we have $\mathrm{Odd}^2 \in \langle\Gamma\rangle_{\not\ni,\neq}$, and for $\langle\Gamma\rangle = \mathrm{II}_2$ we have $\mathrm{C}_0 \times \mathrm{C}_1 \in \langle\Gamma\rangle_{\not\ni,\neq}$, it holds $\mathrm{QBF}_2(\Gamma) \leq_m^{\log} \mathrm{SCEPT}(\Gamma)$. With the Π_2^{P}-hardness of $\mathrm{QBF}_2(\Gamma)$ from [Hem04], it follows that SCEPT(Γ) is hard for coNP. □

We combine the results of this section in the following two theorems. Note that the first theorem classifies the complexity of CRED(Γ) in four different complexity classes, whereas in the classification for SCEPT(Γ) there are only three different complexity classes involved. For a graphical overview of these classifications see Figure 5.3.

Theorem 5.18. *Let* Γ *be a finite constraint language over* $\{0,1\}$*. Then the following holds:*

- *If* $\mathrm{IN}_2 \subseteq \langle\Gamma\rangle$*, then* CRED(Γ) *is* Σ_2^{P}*-complete.*
- *Otherwise, if* $\mathrm{IR}_2 \subseteq \langle\Gamma\rangle$ *or* $\mathrm{ID} \subseteq \langle\Gamma\rangle$*, then* CRED(Γ) *is* NP*-complete.*
- *Otherwise, if* $\mathrm{IN} \subseteq \langle\Gamma\rangle$*, then* CRED(Γ) *is* coNP*-complete.*
- *In all other cases* CRED(Γ) *is decidable in polynomial time.*

Theorem 5.19. *Let* Γ *be a finite constraint language over* $\{0,1\}$*. Then the following holds:*

- *If* $\mathrm{IN}_2 \subseteq \langle\Gamma\rangle$*, then* SCEPT(Γ) *is* Π_2^{P}*-complete.*
- *Otherwise, if* $\mathrm{IR}_2 \subseteq \langle\Gamma\rangle$ *or* $\mathrm{ID} \subseteq \langle\Gamma\rangle$*, then* SCEPT(Γ) *is* coNP*-complete.*
- *In all other cases* SCEPT(Γ) *is decidable in polynomial time.*

We achieved full classifications for the three problems EXT(Γ), CRED(Γ) and SCEPT(Γ) using the weak base method. These classification remain true if we allow the constraint formulas to have existential quantified variables (in this case we speak of conjunctive queries) as was proven in [CHS07].

It is worth noticing that the constraint languages generating one of the Boolean co-clones IN, II, II_0, and II_2 are the only cases, where the complexity of EXT(Γ) is not the same as the complexity CRED(Γ) and where the complexity of SCEPT(Γ) is not dual to the complexity of CRED(Γ). Note that the corresponding cases for credulous and skeptical reasoning are incorrect in [CHS07].

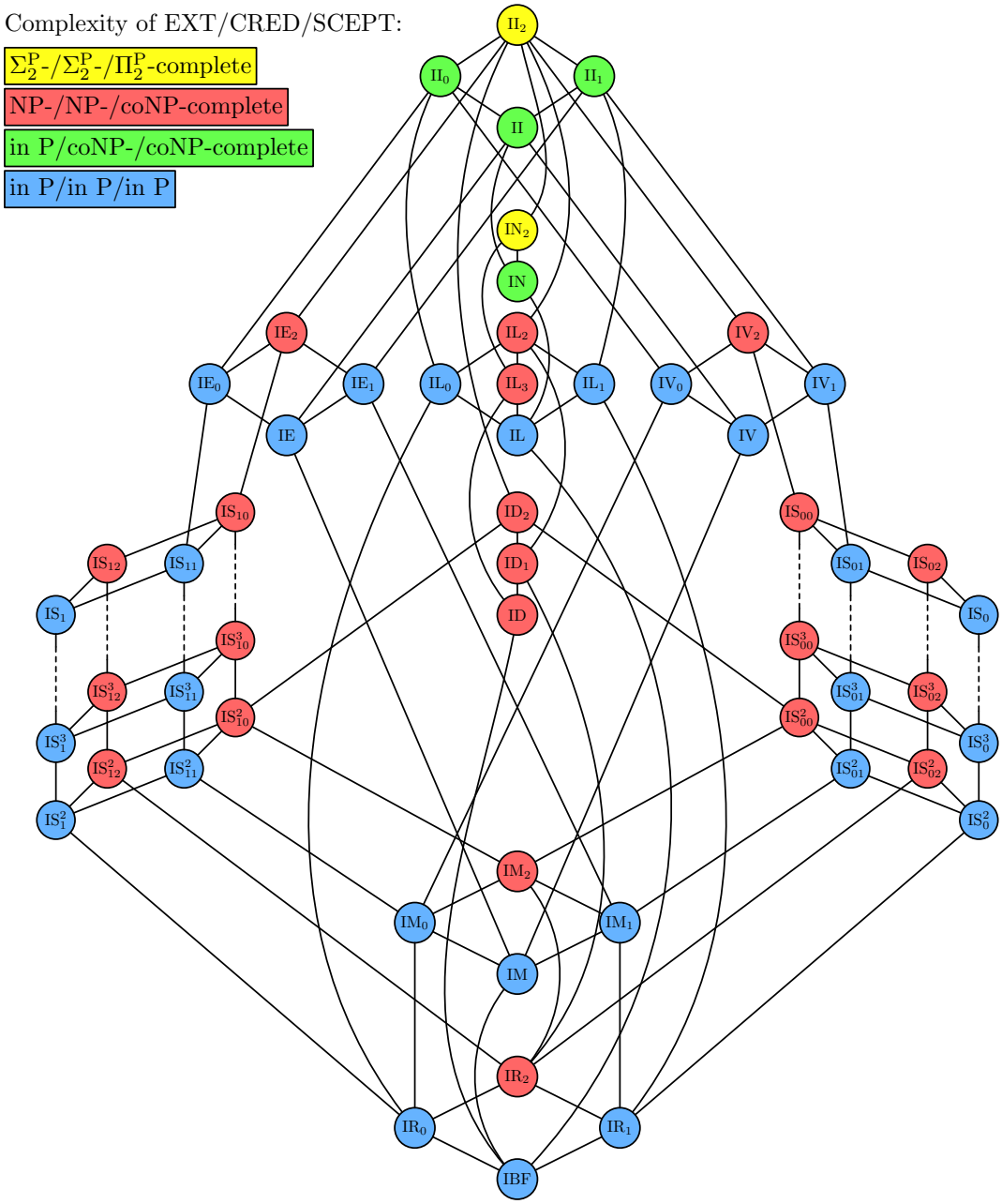

Figure 5.1: The Complexity of Default Reasoning

Chapter 6

The Enumeration Problem

Up to now we studied the complexity of different decision and counting problems for constraint formulas. In this chapter we look at a the *enumeration problem* for constraint formulas, which is the problem to enumerate all solutions of a given constraint formula. This problem is of great relevance, because in practice we are not only interested in knowing whether a formula is satisfiable, we also want to know which solutions the formula has. For example when the formula represents a database query, the solutions represent the matching entries.

In contrast to counting and decision problems, for enumerating problems neither complexity classes nor notions for reductions have been established. We use the notion of *polynomial delay* to express efficient enumerability. Roughly speaking, polynomial delay means we have to wait only polynomial time for the next solution to be printed. Another difference to Chapters 4 and 5, where we only look at the Boolean domains, is that we consider the enumeration problem for constraint languages over arbitrary domains. Although one can show that the existence of efficient enumeration algorithms for Γ-formulas depends only on the weak system generated by Γ, both the lack of established complexity notions and leaving the Boolean domain make it difficult to apply the tools from Chapter 3, because we do not know the structure of the strong partial clones over arbitrary domains.

In fact for the Boolean domain Creignou and Hébrard showed that efficient enumeration algorithms for Γ formulas exist, if and only if the constraint satisfaction problem for constraint formulas over $\Gamma \cup \{\mathrm{C}_0, \mathrm{C}_1\}$ is solvable in polynomial time [CH97]. The algorithm that works in these cases is of a simple type and was generalized to arbitrary domains by Cohen [Coh04]. To our knowledge this elementary algorithm captures all known efficiently enumerable constraint languages to date. We prove that there are more sophisticated algorithms that allow to enumerate solutions efficiently in cases where the well

known algorithm is not efficient. But we will see that these algorithms are limited, when we want to enumerate solutions in a given order.

After presenting basic definitions and previous results in Section 6.1, we give a template for new enumeration algorithms in Section 6.2. Then, in Section 6.3, we give criteria to identify cases where the new algorithms work efficiently. Especially the results in Section 6.3 are very technical, therefore we explain all constructions with an example over the three-element domain.

In Section 6.4 we show that we can efficiently enumerate solutions in a highly customizable order exactly in the cases where the elementary algorithm mentioned above yields efficient enumerability. Finally we consider a fragment of all constraint languages over the three-element domain and identify all its efficiently enumerable cases in Section 6.5.

6.1 Previous Results

How can we measure the complexity of the task to enumerate all solutions of a given formula? Since in general number of solutions for a formula φ is exponential in the number of variables appearing in φ, the time needed to print all solutions does not provide the information if this is an easy task or not. To capture the difficulty of finding the solutions we use the notion of *polynomial delay* introduced by Johnson, Papadimitriou and Yannakakis [JPY88].

For a constraint language Γ an *enumeration algorithm* is an algorithm that, given a Γ-formula φ as input, puts out every solution of φ exactly once.

Definition 6.1. An algorithm has *polynomial delay*, if the computing time before the first output, between every two consecutive outputs, and after the last output until halting is each bounded by a fixed polynomial in the length of the input.

If there is an enumeration algorithm with polynomial delay for Γ, we say that Γ is *efficiently enumerable*. Using a polynomial delay enumeration algorithm we have to wait only polynomial time for either the first solution or, if none exists, the halting of the algorithm, therefore it is trivial that $\mathrm{CSP}(\Gamma) \in \mathrm{P}$, if Γ is efficiently enumerable..

We consider a very simple enumerating scheme suggested by Valiant in [Val79b]. Given a formula φ over the domain D with $\mathrm{Var}\,\varphi = \{x_1, \ldots, x_n\}$ we do the following: for every $a \in D$ we check whether $\varphi \wedge \mathrm{C}_a(x_1)$ is satisfiable and if this is the case we recursively enumerate all solutions I of $\varphi[x_1/a]$, extended by $I(x_n) = a$.

This approach leads to an enumeration algorithm with polynomial delay, if the satisfiability tests can be done in polynomial time, which yields to the

following theorem from Cohen. For a constraint language Γ over a domain D we define $\Gamma^+ =_{def} \Gamma \{\mathrm{C}_a \mid a \in D\}$.

Theorem 6.2 ([Coh04]). *Let Γ be a finite constraint language. If $\mathrm{CSP}(\Gamma^+)$ is in P, then Γ is efficiently enumerable.*

Creignou and Hébrard showed in [CH97] that if we only look at Boolean constraint languages, then the previous theorem captures all efficient cases. Note that, supposing $\mathrm{P} \neq \mathrm{NP}$, a Boolean constraint language Γ is Schaefer if and only if $\mathrm{CSP}(\Gamma^+) \in \mathrm{P}$.

Theorem 6.3 ([CH97]). *Let Γ be a finite constraint language over $\{0,1\}$. Then Γ is efficiently enumerable if and only if $\mathrm{CSP}(\Gamma^+) \in \mathrm{P}$.*

We will see that, when considering other domains than the Boolean, there are are efficiently enumerable constraint languages Γ, such that $\mathrm{CSP}(\Gamma^+)$ is NP-complete and we will develop criteria to identify such constraint languages. To give an intuition for our techniques, we use the following example to explain the constructions in this chapter.

Example 6.4. Let Γ_{ex} be the constraint language over the domain $\{1,2,3\}$, that contains only the following relation.

$$R_{ex} =_{def} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 2 & 0 & 0 & 2 \\ 2 & 0 & 2 & 0 \\ 2 & 2 & 0 & 0 \end{pmatrix}$$

We show that $\Gamma_{ex}^+ = \{R_{ex}, \mathrm{C}_0, \mathrm{C}_1, \mathrm{C}_2\}$ is NP-complete. Since CSP(1-in-3) is NP-complete due to Schaefer's Theorem (see Theorem 2.11), it suffices to show $\mathrm{CSP}(\text{1-in-3}) \leq_m^{\log} \mathrm{CSP}(\Gamma_{ex}^+)$. Let

$$\varphi =_{def} \text{1-in-3}(x_1, y_1, z_1) \wedge \cdots \wedge \text{1-in-3}(x_n, y_n, z_n)$$

and

$$\varphi' =_{def} R_{ex}(v, x_1, y_1, z_1) \wedge \cdots \wedge R_{ex}(v, x_n, y_n, z_n) \wedge \mathrm{C}_2(v),$$

where v is a new variable. It is easy to see that φ has a solution if and only if φ' has one.

Hence $\mathrm{CSP}(\Gamma_{ex}^+)$ is NP-complete. So it seems that the algorithm outlined above cannot be implemented with polynomial delay (unless P = NP) and Theorem 6.2 does not give us that Γ_{ex} is efficiently enumerable. However, in the course of this chapter, we will present an efficient enumeration algorithm for Γ_{ex}.

Although Γ_{ex} will be a counter-example for the converse direction in Theorem 6.2, we will show in section 6.4 that we can generalize Theorem 6.3 to larger domains, by providing an even stronger notion of efficient enumerability, that is equivalent to $\mathrm{CSP}(\Gamma^+) \in \mathrm{P}$.

6.2 Partial Enumerability

The main idea when enumerating the solutions of a Γ_{ex}-formula φ is to divide the problem in two tasks: the first one is to find all "pre-solutions" for φ which map every variable either to 0 or to $\{1, 2\}$. The second task is to refine this "pre-solutions" to "real" solutions by specifying which of the variables associated with $\{1, 2\}$ take the value 1 and which of them take the value 2. An advantage of this approach is, that for finding the "pre-solutions" we do not distinguish between 1 and 2, and for refining the "pre-solutions" we only need to consider the values 1 and 2. So we splitted the problem of enumerating the solutions for a formula over a three-valued domain in two Boolean enumeration tasks. This allows us to apply the classification from Creignou and Hébrard for the Boolean case stated in Theorem 6.3. We need the next definitions to formalize this idea.

Let D be a domain, and E a partition of D. We denote by $\sim_E$ the equivalence relation corresponding to E and by $f^E : D \to E$ the function that maps every element to its equivalence class.

We say a partition E' of D is a refinement of E if for $a, b \in D$ holds that $a \sim_{E'} b$ implies $a \sim_E b$, i.e., if $f^{E'}(a) \subseteq f^E(a)$ for every $a \in D$. In this case we write $E' \leq E$.

Let Γ be a constraint language over D, let φ be a Γ-formula and let $E' \leq E$ be partitions of D. Let $I : \mathrm{Var}(\varphi) \to E$ and $I' : \mathrm{Var}(\varphi) \to E'$ be assignments of E and E' to $\mathrm{Var}(\varphi)$. If for all $x \in \mathrm{Var}(\varphi)$ we have $I'(x) \subseteq I(x)$, then we say I' is *compatible* with I. If φ has a solution $J : \mathrm{Var}(\varphi) \to D$, such that for all $x \in \mathrm{Var}(\varphi)$ it holds $J(x) \in I(x)$, then I is a *partial E-solution* for φ and we say J is *compatible* with I. The set of all partial E-slutions or φ is denoted with $\mathrm{Sol}^E(\varphi)$.

The *discrete partition of D*, where we have exactly one partition for each element, we denote by $D^{disc} =_{def} \{\{a\} \mid a \in D\}$.

Definition 6.5. Let Γ be a constraint language over a domain D and let E_1 and E_2 be partitions of D, such that E_2 is a refinement of E_1.

- Γ is *efficiently E_1-enumerable*, if there exists an algorithm which has polynomial delay and which, given a Γ-formula φ, enumerates $\mathrm{Sol}^{E_1}(\varphi)$.

- Γ is *efficiently $E_1 \to E_2$-enumerable*, if there exists an algorithm which has polynomial delay and which, given a Γ-formula φ and an assignment $I\colon \mathrm{Var}(\varphi) \to E_1$, enumerates all partial E_2-solutions for φ that are compatible with I.

- Γ is *efficiently $E_1 \to D$-enumerable*, if Γ is efficiently $E_1 \to D^{disc}$-enumerable.

Note that partial D^{disc}-solutions correspond directly to "real" solutions, therefore Γ is efficiently D^{disc}-enumerable if and only if Γ is efficiently enumerable.

In the following theorem we show how we can obtain efficient enumeration algorithms by combining efficient algorithms that enumerate partial solutions for different partitions of the domain.

Theorem 6.6. *Let Γ be a constraint language over a domain D, and let E_1 and E_2 be partitions of D such that $E_2 \leq E_1$. If Γ is efficiently E_1-enumerable and efficiently $E_1 \to E_2$-enumerable, then Γ is efficiently E_2-enumerable.*

Proof. Let A be an algorithm that, given a Γ-formula φ, enumerates $\mathrm{Sol}^{E_1}(\varphi)$ with polynomial delay, and let B be an algorithm that, given a Γ-formula φ and an assignment $I : \mathrm{Var}(\varphi) \to E_1$, enumerates all partial E_2-solutions compatible to I with polynomial delay. Note that both algorithms exist because Γ is efficiently E_1-enumerable and efficiently $E_1 \to E_2$-enumerable.

We modify A: let φ be the input of A. Every time A wants to print a partial E_1-solution I, we call instead B on φ and I.

Obviously the new algorithm has polynomial delay, because both algorithms A and B have polynomial delay, and for each of the solutions $I \in \mathrm{Sol}^{E_1}(\varphi)$, there is a compatible partial E_2-solution for φ which is put out by B.

Since all printing is done by B, it holds that every output is a partial E_2-solution. Note that for two different partial E_1-solutions I, I' there is no partial E_2 solution compatible with both I and I', therefore no output is printed twice.

Finally every partial E_2-solution for φ is printed out, because every partial E_2-solution I is compatible with the partial E_1-solution J, which is defined uniquely by $J(a) \supseteq I(a)$ for every $a \in D$. Thus, it follows that Γ is efficiently E_2-enumerable. □

For the constraint language Γ_{ex} from Example 6.4 we consider the partition $E = \{\{0\}, \{1, 2\}\}$. The "pre-solutions" we spoke of in the beginning of this section correspond to partial E-solutions. In the next section we will see that Γ_{ex} is efficiently E-enumerable and efficiently $E \to \{0, 1, 2\}$-enumerable. So,

according to the previous theorem we can build an efficient enumeration algorithm for Γ by nesting the efficient algorithms that exist due to the efficient E- and $E \to \{0,1,2\}$-enumerability of Γ_{ex}.

The following corollary follows directly from Theorem 6.6.

Corollary 6.7. *Let Γ be a constraint language over a domain D and let $E_k \leq \cdots \leq E_1$ be partitions of D such that Γ is efficiently E_1-enumerable and efficiently $E_i \to E_{i+1}$-enumerable for every $i \in \{1, \ldots, k-1\}$. Then Γ is efficiently enumerable.*

6.3 Criteria

In this section we develop criteria for partial enumerability. We construct constraint languages at which we have to look if we want to know if a given constraint language is efficiently E- or $E_1 \to E_2$-enumerable.

Let D and D' be domains, let Γ be a constraint language over D and let $f : D \to D'$ be a unary function. For a relation R from Γ, we define $R_f =_{def} \{(f(a_1), \ldots, f(a_n)) \mid (a_1, \ldots, a_n) \in R\}$ and further we define $\Gamma_f =_{def} \{R_f \mid R \in \Gamma\}$.

Let E be a partition of a domain D. A function $g : D \to D$ is a *representation function* for E if for all $a, b \in D$ it holds that $a \sim_E b$ implies $g(a) = g(b)$, and that $g(a) \in f^E(a)$. That means g maps every element from D to a unique representative of its equivalence class with respect to $\sim_E$.

We now are ready to present our criterion for efficient E-enumerability. Note that for a partition E of D and a constraint language Γ over D, it holds that Γ_{f^E} is a relation over the domain E.

Theorem 6.8. *Let Γ be a finite constraint language over a domain D and let E be a partition of D such that there exists a representation function for E which is a polymorphism of Γ. Then the following is equivalent:*

1. *Γ is efficiently E-enumerable,*
2. *Γ_{f^E} is efficiently enumerable.*

Proof. Let φ be a Γ-formula and let φ' be the Γ_{f^E}-formula that has a clause $R_{f^E}(x_1, \ldots, x_n)$ if and only if φ has the clause $R(x_1, \ldots, x_n)$. That means φ' can be obtained from φ in polynomial time by, for every $R \in \Gamma$, replacing each R-clause by an R_{f^E}-clause with the same variables, and vice versa.

We show that $\mathrm{Sol}^E(\varphi) = \mathrm{Sol}(\varphi')$, then the theorem follows because we can enumerate the partial E-solutions of φ by enumerating the solutions of φ' and the other way round.

Let $I : \mathrm{Var}(\varphi) \to E$ be an assignmet of E to the variables of Γ. We show that I is a partial E-solution for φ if and only if I is a solution for φ'.

First assume that I is a partial E-solution for φ. This means there is a solution $J : \mathrm{Var}(\varphi) \to D$ for φ such that for every $a \in D$ we have $J(a) \in I(a)$. Let $R_{f^E}(x_1, \ldots, x_n)$ be a clause in φ'. Then $(J(x_1), \ldots, J(x_n)) \in R$, because $R(x_1, \ldots, x_n)$ is a clause in φ. Due to the definition of R_{f^E}, it follows

$$(f^E(J(x_1)), \ldots, f^E(J(x_n))) \in R_{f^E}.$$

Since f^E maps every element of D to its equivalence class, it holds $f^E \circ J = I$. Therefore we have that I is a solution of φ'.

Now we assume that I is a solution for φ. Let $R(x_1, \ldots, x_n)$ a clause from φ. Since $R_{f^E}(x_1, \ldots, x_n)$ is a clause in φ' it holds that $(I(x_1), \ldots, I(x_n)) \in R_{f^E}$. Due to the construction of R_{f^E}, it follows that there exists a tuple $(a_1, \ldots, a_n) \in R$ such that

$$(f^E(a_1), \ldots, f^E(a_n)) = (I(x_1), \ldots, I(x_n)).$$

Due to the prerequisites, there exists a $g \in \mathrm{Pol}(\Gamma)$ which is a representation function for E. We define an assignment $J : \mathrm{Var}(\varphi) \to D$ by setting for every $x \in \mathrm{Var}(\varphi)$: $J(x) =_{def} g(a)$ if and only if $a \in I(x)$. Note that J is well defined because g is a representation function for E. It follows that $(J(x_1), \ldots, J(x_n)) = (g(a_1), \ldots, g(a_n))$. Since $g \in \mathrm{Pol}(\Gamma)$ we have $(J(x_1), \ldots, J(x_n)) \in R$, therefore J is a solution for φ. Due to the construction of J it holds for every $a \in D$ that $J(a) \in I(a)$ is true. Hence I is a partial E-solution for φ.

Thus it holds $\mathrm{Sol}^E(\varphi) = \mathrm{Sol}(\varphi')$, which completes the proof. □

We consider the constraint language Γ_{ex} from Example 6.4 again. If we choose the partition $E = \{\{0\}, \{1, 2\}\}$, then $f_{2\to1} : \{0, 1, 2\} \to \{0, 1, 2\}$, defined $f_{2\to1}(2) =_{def} 1$ and $f_{2\to1}(a) =_{def} a$ for $a \in \{0, 1\}$, is a representation function for E. It can easily be verified that $f_{2\to1} \in \mathrm{Pol}(\Gamma_{ex})$, therefore it follows from Theorem 6.8 that Γ_{ex} is efficiently E-enumerable if and only if $\Gamma_{ex\,f^E}$ is efficiently enumerable. It holds that $\Gamma_{ex\,f^E} = \left\{R_{ex\,f^E}\right\}$ and

$$R_{ex\,f^E} = \begin{pmatrix} \{1,2\} & \{0\} & \{0\} & \{0\} \\ \{1,2\} & \{0\} & \{0\} & \{1,2\} \\ \{1,2\} & \{0\} & \{1,2\} & \{0\} \\ \{1,2\} & \{1,2\} & \{0\} & \{0\} \end{pmatrix}$$

To get a Booelan relation we identify $\{0\}$ with 0 and $\{1, 2\}$ with 1:

$$R_{ex\,f^E} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

It can be seen that $\wedge$ is a polymorphism of this relation, therefore it holds that $\Gamma_{ex f^E}$ is Schaefer and can be enumerated efficiently due to Theorem 6.3. Hence, Γ_{ex} is efficiently E-enumerable.

We need the next definition to be able to state the criterion for efficient $E_1 \to E_2$-enumerability. For an n-ary relation R over a domain D and for a set of indices $I \subseteq \{1, \ldots, n\}$ we define the *projection of R to I* to be the relation $R_I =_{def} \{(a_{i_1}, \ldots, a_{i_k}) \mid (a_1, \ldots, a_n) \in R\}$, where $I = \{i_1, \ldots, i_k\}$ and $i_1 < \cdots < i_k$.

Definition 6.9. Let D be a domain and E_1 and E_2 partitions of D such that $E_2 \leq E_1$.

1. For an n-ary relation R over D and a tuple $v \in R$ we define:

$$R_v^{E_1 \to E_2} =_{def} \left\{ f^{E_2}(t) \mid t \in R \text{ and } v = f^{E_1}(t) \right\}_{\{i \in \{1,\ldots,n\} \mid v[i] \notin E_2\}}$$

2. For a constraint language Γ over D we define:

$$\Gamma^{E_1 \to E_2} =_{def} \left\{ R_v^{E_1 \to E_2} \mid R \in \Gamma \text{ and } v \in R_{f^{E_1}} \right\}$$

Observe that $\Gamma^{E_1 \to E_2}$ is a constraint language over the domain E_2.

Theorem 6.10. *Let Γ be a finite constraint language over a domain D, and let $E_2 \leq E_1$ be partitions of D, such that there exists a representation function for E_2 which is a polymorphism of Γ. Then the following is equivalent:*

1. *Γ is efficiently $E_1 \to E_2$-enumerable,*
2. *$\Gamma^{E_1 \to E_2}$ is efficiently enumerable.*

Proof. Let $\Gamma^{E_1 \to E_2}$ be efficiently enumerable. We show that Γ is efficiently $E_1 \to E_2$-enumerable. We need to show that there is an algorithm that takes a Γ-formula φ and an assignment $I : \mathrm{Var}(\varphi) \to E_1$ of E_1-to the variables of φ, and that enumerates all partial E_2-solutions for φ that are compatible to I with polynomial delay. The algorithm works as follows: first we check for every clause $R(x_1, \ldots, x_n)$ in φ if $(I(x_1), \ldots, I(x_n)) \in R_{f^{E_1}}$. If there is a clause for which this is not the case, we halt, because it follows that no solution for φ is compatible to I, and therefore no partial E_2-solution of φ is either.

Otherwise we construct the $\Gamma^{E_1 \to E_2}$-formula ψ that has for every clause $R(x_i, \ldots, x_n)$ from φ the corresponding clause $R_v^{E_1 \to E_2}(x_{i_1}, \ldots, x_{i_k})$, where $v = (I_1(x_1), \ldots, I_1(x_n))$ and $\{i_1, \ldots, i_k\} = \{i \in \{1, \ldots, n\} \mid I_1(x_i) \notin E_2\}$ such that $i_1 < \cdots < i_k$. Clearly this construction can be done in polynomial time.

Now we use a polynomial delay enumeration algorithm for $\Gamma^{E_1 \to E_2}$ to enumerate the solutions for ψ. Before printing a solution $J : \mathrm{Var}(\psi) \to E_2$ for φ, we extend it to $\mathrm{Var}(\varphi)$ by setting $J(x) =_{def} I(x)$ for every $x \in \mathrm{Var}(\varphi) \setminus \mathrm{Var}(\psi)$.

Since $\Gamma^{E_1 \to E_2}$ is efficiently enumerable, this algorithm can be implemented to work with polynomial delay. We show the correctness of the algorithm in the end of the proof.

Let Γ be efficiently $E_1 \to E_2$-enumerable. We show that there is an algorithm that enumerates all solutions of a given $\Gamma^{E_1 \to E_2}$-formula ψ with polynomial delay. First we check if for every variable $x \in \mathrm{Var}(\psi)$ the following holds: if x appears twice in ψ, lets say x is the i-th variable in an R-clause and the j-th variable in an S-clause for some not necessarily distinct relations $R, S \in \Gamma^{E_1 \to E_2}$, then the set R^i of all values in the i-th column of R and the set S^j of all values in the j-th column of S are not disjoint. If that is not the case, then it is obvious that ψ is not satisfiable, which means we let the algorithm halt.

Otherwise we construct a Γ-formula φ and an assignment $I : \mathrm{Var}(\varphi) \to E_1$ in the following way: for every clause $R_v^{E_1 \to E_2}(x_{i_1}, \ldots, x_{i_k})$ with $R \in \Gamma$, $v \in R_{f^{E_1}}$, and $\{i_1, \ldots, i_k\} = \{i \in \{1, \ldots, n\} \mid v[i] \notin E_2\}$ such that $i_1 < \cdots < i_k$, we add the clause $R(x_1, \ldots, x_n)$ to ψ, where every variable x_i with $v[i] \in E_2$ is a new variable. Additionally we set $I(x_i) =_{def} v[j]$. Note that I is well defined, because we made sure that all columns of relations that correspond to the same variable are not disjoint. Due to the construction of $R_v^{E_1 \to E_2}$, this implies that the corresponding values in the tuple v are equal. This construction is reverse to the one in the first part of the proof and can be performed in polynomial time.

Now we enumerate all partial E_2-solutions J for φ that are compatible to I. For every enumerated J we print the restriction $J|_{\mathrm{Var}(\psi)}$ to the variables of φ. Since Γ is efficiently $E_1 \to E_2$-enumerable, the algorithm works with polynomial delay.

To show the correctness of both exhibited algorithms, it suffices to prove that every partial E_2-solution of φ which is compatible with I can be restricted to a solution of ψ, and every solution of ψ can be extended to exactly one partial E_2-solution of φ compatible with I.

First let $J : \mathrm{Var}(\varphi) \to E_2$ be a partial E_2-solution for φ that is compatible with I. Then there exists a solution J' for φ such that $J'(x) \in J(x) \subseteq I(x)$ for every $x \in \mathrm{Var}(\varphi)$. That means it holds for every clause $R(x_1, \ldots, x_n)$ from φ and the corresponding clause $R_v^{E_1 \to E_2}(x_{i_1}, \ldots, x_{i_k})$ from ψ that

$$(J(x_1), \ldots, J(x_n)) \in \left\{ f^{E_2}(t) \mid t \in R \text{ and } v = f^{E_1}(t) \right\}.$$

Note that $v = (I(x_1), \ldots, I(x_n))$. It follows that $(J(x_{i_1}), \ldots, J(x_{i_k})) \in R_v^{E_1 \to E_2}$, therefore J restricted to $\mathrm{Var}(\psi)$ is a solution for ψ.

For the other direction let $J : \mathrm{Var}(\psi) \to E_2$ be a solution for ψ. We define $J' : \mathrm{Var}(\varphi) \to E_2$ by $J'(x) = J(x)$ for $x \in \mathrm{Var}(\psi)$ and $J'(x) = I(x)$ otherwise (note that $I(x) \in E_2$ in these cases and therefore J' is the only extension of J compatible with I). Then for every clause $R(x_1, \ldots, x_n)$ from φ and the corresponding clause $R_v^{E_1 \to E_2}(x_{i_1}, \ldots, x_{i_k})$ from ψ it holds that

$$(J'(x_{i_1}), \ldots, J'(x_{i_k})) = (J(x_{i_1}), \ldots, J(x_{i_k})) \in R_v^{E_1 \to E_2}.$$

It follows that $(J'(x_1), \ldots, J'(x_n)) \in \left\{ f^{E_2}(t) \mid t \in R \text{ and } v = f^{E_1}(t) \right\}$ and J' is compatible to I. Let t be a tuple from R such that $f^{E_2}(t) = (J'(x_1), \ldots, J'(x_n))$. Let g be the representation function for E_2 which is an element of $\mathrm{Pol}(\Gamma)$. Then we know that $g(t) \in R$. Therefore the assignment $K : \mathrm{Var}(\varphi) \to D$ defined by $K(x) =_{def} g(a)$ if $a \in J'(x)$ is a solution for φ that is compatible with J'. Hence J' is a partial E_2-solution for φ that is compatible with I. That proves the correctness of the two algorithms above. □

We use the previous theorem to show that Γ_{ex} from Example 6.4 is efficiently $E \to D$-enumerable for $D = \{0, 1, 2\}$, and $E = \{\{0\}, \{1, 2\}\}$. First we construct the constraint language $\Gamma_{ex}^{E \to D}$. We have seen above that

$$\begin{aligned} v_1 &= (\{1,2\}, \{0\}, \{0\}, \{0\}), \\ v_2 &= (\{1,2\}, \{0\}, \{0\}, \{1,2\}), \\ v_3 &= (\{1,2\}, \{0\}, \{1,2\}, \{0\}), \quad \text{and} \\ v_4 &= (\{1,2\}, \{1,2\}, \{0\}, \{0\}) \end{aligned}$$

are all tuples from $R_{ex f^E}$. Therefore ${\Gamma_{ex}}^{E \to D}$ consists of the following relations:

$$\begin{array}{lcll} R_{ex\,v_1}^{E \to D} & = & \{(1,0,0,0)\}_{\{1\}} & = \{(1)\} \\ R_{ex\,v_2}^{E \to D} & = & \{(1,0,0,1),(2,0,0,2)\}_{\{1,4\}} & = \{(1,1),(2,2)\} \\ R_{ex\,v_3}^{E \to D} & = & \{(1,0,1,0),(2,0,2,0)\}_{\{1,3\}} & = \{(1,1),(2,2)\} \\ R_{ex\,v_4}^{E \to D} & = & \{(1,1,0,0),(2,2,0,0)\}_{\{1,2\}} & = \{(1,1),(2,2)\} \end{array}$$

If we identify the value 2 with 0, then ${\Gamma_{ex}}^{E \to D}$ is invariant under $\wedge$, therefore ${\Gamma_{ex}}^{E \to D}$ is Schaefer. Due to Theorem 6.3 it holds that ${\Gamma_{ex}}^{E \to D}$ is efficiently enumerable and then it follows from Theorem 6.10 that Γ_{ex} is efficiently $E \to D$-enumerable.

We showed above that Γ_{ex} is efficiently E-enumerable, therefore it follows from Theorem 6.6 that Γ_{ex} is efficiently enumerable. Since $\mathrm{CSP}(\Gamma_{ex}^{+})$ is NP-complete, we obtain the following corollary.

Corollary 6.11. *There exists an efficiently enumerable constraint language* Γ *over the three-element domain such that* $\mathrm{CSP}(\Gamma^{+})$ *is* NP*-complete.*

6.4 Lexicographical Orderings

It can be seen easily that the algorithm outlined in Section 6.1 orders its output by grouping together all solutions that map x_1 to the same value.

We define a stronger notion of efficient enumerability in which we require to print the solutions in a given order.

A *variable E-lexicographical enumeration algorithm for* Γ is an algorithm which takes a Γ-formula φ, a linear order $<_{\mathrm{Var}(\varphi)}$ on $\mathrm{Var}(\varphi)$, and for every $x \in \mathrm{Var}(\varphi)$ a linear order $<_x$ on E as input and enumerates all partial E-solutions of φ in the following linear order: if $I : \mathrm{Var}(\varphi) \to E_2$ and $I' : \mathrm{Var}(\varphi) \to E_2$ are partial E_2-solutions for φ, then I is printed before I' if and only if there is some $x \in \mathrm{Var}(\varphi)$ such that for all $y \in \mathrm{Var}(\varphi)$ with $y \leq_{\mathrm{Var}(\varphi)} x$ it holds that $I(y) = I'(y)$ and $I(x) <_x I'(x)$.

That means the algorithm prints the solutions in a lexicographical order, where each variable x has its own sorting criterion $<_x$. In the context of databases it is very natural to demand the output of a query to be ordered by different criteria for different columns. For example we would like to order the movies of the last year descending in the number of awards and, in the case of ties, assending in the number of tickets sold.

We say that Γ is efficiently variable lexicographical (E-lexicographical) enumerable if Γ has a variable D^{disc}-lexicographical (E-lexicographical) enumeration algorithm with polynomial delay.

The next theorem shows, that a constraint language is efficiently variable lexicographical enumerable if and only if a modification of the algorithm outlined in Section 6.1 works. Note that every class of a partition E of D can be seen as unary relation over D.

Theorem 6.12. *Let Γ be a constraint language over a domain D and let E be a partition on D. Then Γ is efficiently variable E-lexicographical enumerable if and only if* $\mathrm{CSP}(\Gamma \cup E)$ *is in* P.

Proof. Let $E = \{D_1, \ldots, D_k\}$, and first assume that $\mathrm{CSP}(\Gamma \cup E)$ is decidable in polynomial time. Let φ be a Γ-formula, let $<_{\mathrm{Var}(\varphi)}$ be a linear order on $\mathrm{Var}(\varphi)$ and let $<_x$ be a linear order on E for every $x \in \mathrm{Var}(\varphi)$. We give an algorithm that takes a Γ-formula φ, the above orders, and a $\{D_1, \ldots, D_k\}$-formula ψ with $\mathrm{Var}\,\psi \subseteq \mathrm{Var}\,\varphi$ as input and generates all partial solutions of $\varphi \wedge \psi$ in the correct order with polynomial delay. To enumerate all partial solutions of φ, apply the algorithm on φ and the empty formula ψ.

1: **procedure Generate**
2: **input** a Γ-formula φ, an E-formula ψ with $\mathrm{Var}(\psi) \subseteq \mathrm{Var}(\varphi)$, and linear orders $<_{\mathrm{Var}(\varphi)}$ on $\mathrm{Var}(\varphi)$ and $<_x$ on E for all $x \in \mathrm{Var}(\varphi)$

3: **if** $\varphi \wedge \psi \notin$ SAT **then**
4: **halt**
5: **end if**
6: **if** $\mathrm{Var}(\varphi) = \mathrm{Var}(\psi)$ **then**
7: Print the partial assignment uniquely defined by ψ
8: **halt**
9: **end if**
10: Let x be the $<_{\mathrm{Var}(\varphi)}$-smallest variable in $\mathrm{Var}(\varphi) \setminus \mathrm{Var}(\psi)$
11: Let $D_{i_1} <_x D_{i_2} <_x \cdots <_x D_{i_k}$
12: **for** $j = 1$ to $j = k$ **do**
13: **Generate**$(\varphi,\ \psi \wedge D_{i_j}(x),\ <_{\mathrm{Var}(\varphi)},\ <_x$ for all $x \in \mathrm{Var}(\varphi))$
14: **end for**

This algorithm is a generalization of the algorithms in the proof of Theorem 9 in [Coh04] and in Section 3 of [CH97]. It works with polynomial delay, since the satisfiability test used in the algorithm can be performed in polynomial time by our prerequisites.

Now assume that Γ has a variable E-lexicographical enumeration algorithm, and let φ be a $\Gamma \cup E$-formula. For $1 \leq i \leq k$ let

$$X_i =_{def} \{x \in \mathrm{Var}(\varphi) \mid D_i(x) \text{ is a clause in } \varphi\}$$

be the set of all variables that appear in a D_i-clause. Further let $X := \bigcup_{i=1}^{k} X_i$ be the set of variables that are constrained by some D_i from E.

Now, define $<_{\mathrm{Var}(\varphi)}$ to be a linear order on $\mathrm{Var}(\varphi)$ such that for all $x \in X$ and $y \in \mathrm{Var}(\varphi) \setminus X$, it holds that $x <_{\mathrm{Var}(\varphi)} y$. For each $x \in X_i$, define the order $<_x$ in such a way that D_i is the smallest element of E with respect to $<_x$. Let φ' be the Γ-formula obtained from φ by deleting all E-clauses. Now, enumerate the solutions of φ' according to the order as defined above. By construction of the order, the formula φ is satisfiable if and only if the first partial solution I returned by the algorithm satisfies the φ. This gives a polynomial-time decision procedure for $\mathrm{CSP}(\Gamma \cup E)$. □

Observe that $\Gamma \cup D^{disc}$ equals Γ^+, therefore the previous theorem directly leads to the following corollary, which generalizes Theorem 6.3 from Creignou and Hébrard to non-Boolean domains.

Corollary 6.13. *Let Γ be a constraint language over a domain D. Then Γ is efficiently variable lexicographical enumerable if and only if* $\mathrm{CSP}(\Gamma^+) \in \mathrm{P}$.

6.5 Towards a Dichotomy for Three-Element Domains

It is a long-term goal to find a characterization that separates all constraint languages that are efficiently enumerable from those that cannot be enumerated in an efficient way. One challenging aspect in this ambition is, that it is not sufficient to consider only the co-clones to achieve a full classification. Since equivalent formulas have the same set of solutions, it is obvious that the $\langle . \rangle_{\not\equiv,\neq}$-closure provides efficient enumerability and it is not hard to show that the $\langle . \rangle_{\not\equiv}$-closure also has this property. But, in contrast to the question if a formula has a balanced solution and the problems in the context of default reasoning studied in Chapters 4 and 5, there are constraint languages generating the same co-clone, such that one is efficiently enumerable and the other is not (if P $\neq$ NP). An example for those constraint languages over a three-element domain can be found in [SS06a].

However, we look at a fragment of all constraint languages over the three-element domain and identify all of its efficiently enumerable cases. For this classification we need the following definitions and the subsequent lemma.

Let D be a domain an D' a subset of D. A function $f : D^n \to D$ is *conservative* on D', if for all $a_1, \ldots, a_n \in D'$ it holds $f(a_1, \ldots, a_n) \in D'$.

For a domain D and $a, b \in D$, we define $f_{a \to b} : D \to D$ to be the function that fulfills $f(a) = b$ and $f(x) = x$ for all $x \in D \setminus \{a\}$.

Lemma 6.14. *Let Γ be a relation over the domain $D = \{a, b, c\}$, and let $E = \{\{a, b\}, \{c\}\}$ be a partition of D, such that all polymorphisms of Γ are conservative on the classes in E. Then*

$$\{a, b\} \times \mathrm{C}_b \times \mathrm{C}_c \in \langle \Gamma \rangle_{\not\equiv,\neq} \quad \textit{or} \quad \{a, b\} \times \mathrm{C}_c \in \langle \Gamma \rangle_{\not\equiv,\neq}.$$

Proof. To implement the relations we define the following Γ-formula which uses variables from $\{x_a, x_b, x_c\}$:

$$\varphi =_{def} \bigwedge_{R \in \Gamma} \bigwedge_{t \in R} R(x_{t[1]}, \ldots, x_{t[n_R]}),$$

where for every $R \in \Gamma$ we denote by n_R the arity of R.

Before we prove the lemma we show that $f : D \to D$ is a polymorphism of Γ if and only $I_f : \mathrm{Var}(\varphi) \to D$ defined by $I_f(x_\alpha) = f(\alpha)$ for each $\alpha \in D$ is a solution for φ. Let f be a polymorphism of R. By construction of φ, it is obvious that I_{id}, is a solution of φ. Since $f \in \mathrm{Pol}(\Gamma)$, we know that $f(I_{id})$ is a solution of φ as well. Now for $\alpha \in D$, it holds that $f(I_{id})(x_\alpha) = f(I_{id}(x_\alpha)) = f(\alpha)$, i.e., $f(I_{id}) = I_f$ satisfies φ.

For the converse, assume that I_f is a solution of φ. Let $R \in \Gamma$ and let $t \in R$. We show that $f(t) \in R$. Since I_f satisfies φ, we know that I_f satisfies the clause $R(x_{t[1]}, \ldots, x_{t[n_R]})$, i.e., $(I_f(x_{t[1]}), \ldots, I_f(x_{t[n_R]})) \in R$. Obviously, it holds $(I_f(x_{t[1]}), \ldots, I_f(x_{t[n_R]})) = f(t)$, therefore f is a polymorphism of Γ.

To prove the lemma we distinguish two cases:

Case 1: $f_{b\to a} \notin \mathrm{Pol}(\Gamma)$. We prove that $\{a,b\} \times \mathrm{C}_b \times \mathrm{C}_c \in \langle\Gamma\rangle_{\not\exists,\neq}$ by showing that $I : \{x_a, x_b, x_c\} \to \{a,b,c\}$ is a solution of φ if and only if $I(x_a) \in \{a,b\}$, $I(x_b) = b$, and $I(x_c) = c$. First, let I be a solution for φ. Because all polymorphisms of Γ are conservative on $\{a,b\}$ and on $\{c\}$ we know, according to the above, that that $I(x_c) = c$, and $I(x_a), I(x_b) \in \{a,b\}$. Assume that $I(x_b) \neq b$, i.e., $I(x_b) = a$. If $I(x_a) = a$, then, due to the above, it holds that $f_{b\to a}$ is a polymorphism of Γ, which is a contradiction to our assumption. For the other case, if $I(x_a) = b$, then we know that f' is a polymorphism of Γ, where f' is defined by $f'(a) =_{def} b$, $f'(b) =_{def} a$, and $f'(c) =_{def} c$. Since $f_{a\to b} \in \mathrm{Pol}(\Gamma)$, we conclude that $f' \circ f_{a\to b} = f_{b\to a} \in \mathrm{Pol}(\Gamma)$, which again is a contradiction.

Now, let $I(x_a) \in \{a,b\}$, $I(x_b) = b$, and $I(x_c) = c$. It is obvious that $I(x_a) = a$ implies that I is a solution for φ: this follows from the above, because id is always a polymorphism. Therefore, assume that $I(x_a) = b$. Then I corresponds to the function $f_{a\to b}$, which is a polymorphism of Γ. Due to the above it follows that I is a solution for φ.

Case 2: $f_{b\to a} \in \mathrm{Pol}(R)$. We consider the formula ψ which is obtained from φ by identifying the variables x_a and x_b. Then $\mathrm{Var}(\psi) = \{x, x_c\}$, where $x = x_a = x_b$. We show that $\{a,b\} \times \mathrm{C}_c \in \langle\Gamma\rangle_{\not\exists,\neq}$ by proving that $I : \{x, x_c\} \to \{a,b,c\}$ is a solution for ψ if and only if $I(x) \in \{a,b\}$ and $I(x_c) = c$.

First, let I be a solution for ψ. Due to the construction of ψ, it follows from the above, that f defined by $f(a) =_{def} f(b) =_{def} I(x)$, $f(c) =_{def} c$ is a polymorphism of Γ. Since all polymorphisms are conservative on $\{a,b\}$ and on $\{c\}$, it follows that $I(x) \in \{a,b\}$, and $I(x_c) = c$.

Now, assume $I(x) \in \{a,b\}$ and $I(x_c) = c$. Since $f_{a\to b} \in \mathrm{Pol}(R)$, it follows from the above, that I_1 defined by $I_1(x_c) =_{def} c, I_1(x_b) =_{def} I_1(x) =_{def} b$ is a solution for φ. Therefore, I_1 restricted to $\{x, x_c\}$ is a solution for ψ. The second possible assignment fulfilling the requirements is I_2, defined by $I_2(x) =_{def} a, I_2(x_c) =_{def} c$. We extend I_2 to an assignment for φ by setting $I_2(x_a) =_{def} I_2(x_b) =_{def} a$. This is a solution of φ, because $f_{b\to a}$ is a polymorphism of Γ. Hence, $\{a,b\} \times \mathrm{C}_c \in \langle\Gamma\rangle_{\not\exists,\neq}$. □

The following proposition follows directly from [BKJ00].

Proposition 6.15 ([BKJ00])**.** *Let* Γ *be a constraint language over a domain* D*. If every unary polymorphism of* Γ *is a permutation on* D*, then* $\mathrm{CSP}(\Gamma) \in \mathrm{P}$ *if and only if* $\mathrm{CSP}(\Gamma^+) \in \mathrm{P}$*.*

We can identify all efficiently enumerable constraint language over a three-element domain D, whose polymorphisms are conservative on the classes of some non-trivial partition of D. The classification is a corollary of the results in this chapter.

Corollary 6.16. *Let* Γ *be a relation over the domain* $\{a, b, c\}$*, and let* $E = \{\{a, b\}, \{c\}\}$ *be a partition of* D*, such that all polymorphisms of* Γ *are conservative on the classes in* E*. Then the following holds:*

- *If* $f_{a\to b} \notin \mathrm{Pol}(\Gamma)$ *and* $f_{b\to a} \notin \mathrm{Pol}(\Gamma)$*, then* Γ *is efficiently enumerable if and only if* $\mathrm{CSP}(\Gamma) \in \mathrm{P}$*.*
- *Otherwise,* Γ *is efficiently enumerable if and only if* $\Gamma^{E\to D}$ *is Schaefer, and* Γ_{f^E} *is Schaefer (or* $\mathrm{P} = \mathrm{NP}$*).*

Proof. First, assume that $f_{a\to b}$ and $f_{b\to a}$ are not polymorphisms of Γ. Let $f : D \to D$ be a polymorphism of Γ. Since f is conservative on $\{a, b\}$ and on $\{c\}$, it holds that $f(c) = c$ and $f(a), f(b) \in \{a, b\}$. It follows that $f(a) \neq f(b)$, otherwise f would be equal to either $f_{a\to b}$ or $f_{b\to a}$. Therefore f is a permutation on D and due to Proposition 6.15 we know, that $\mathrm{CSP}(\Gamma) \in \mathrm{P}$ if and only if $\mathrm{CSP}(\Gamma^+) \in \mathrm{P}$. Then it follows with Theorem 6.2 that $\mathrm{CSP}(\Gamma) \in \mathrm{P}$ implies that Γ is efficiently enumerable. Conversely, if Γ is efficiently enumerable, then $\mathrm{CSP}(\Gamma)$ is obviously in P.

Now, assume that, without loss of generality, $f_{a\to b} \in \mathrm{Pol}(R)$. Note that $f_{a\to b}$ is a representation function for E. If $\Gamma^{E\to D}$ is Schaefer, and Γ_{f^E} is Schaefer, then both, $\Gamma^{E\to D}$ and Γ_{f^E} are efficiently enumerable according to Theorem 6.3. It follows from Theorem 6.10 that Γ is efficiently $E \to D$-enumerable, and from Theorem 6.8 that Γ is efficiently E-enumerable. Then Γ is efficiently enumerable due to Theorem 6.6.

If on the other hand Γ is efficiently enumerable, then $\mathrm{CSP}(\Gamma) \in \mathrm{P}$. Since every polymorphism of Γ is conservative on the classes of E, and since the constraint language E is invariant exactly under the the functions that are conservative on the classes of E, it follows from Proposition 2.8 that $\langle \Gamma \cup E \rangle = \langle \Gamma \rangle$. Therefore it holds that $\mathrm{CSP}(\Gamma \cup E) \in \mathrm{P}$ and, according to Theorem 6.12, Γ is efficiently E-enumerable. Since $f_{a\to b}$ is a representation function for E we know from Theorem 6.8 that Γ_{f^E} is efficiently enumerable. Then, unless $\mathrm{P} = \mathrm{NP}$, it holds due to Theorem 6.3 that Γ_{f^E} is Schaefer.

We now show that $\Gamma^{E\to D}$ is Schaefer. It is sufficient to show that Γ is efficiently $E \to D$-enumerable, then the result follows from Theorem 6.10 and

Theorem 6.3. Let φ be a Γ-formula and $I : \mathrm{Var}(\varphi) \to E$ an assignment of E to $\mathrm{Var}(\varphi)$. We enumerate all solutions for φ that are compatible I in the following way: if I maps every variable to c, then we test if I is a solution for φ and print I if this is the case; then we halt, because I cannot be refined further. Otherwise we use Lemma 6.14. If

$$S =_{def} \{a, b\} \times \mathrm{C}_b \times \mathrm{C}_c \in \langle\Gamma\rangle_{\not\equiv,\neq},$$

there is according to Corollary 2.10 a Γ-formula ψ_S such that $\psi_S(x, y, z) \equiv S(x, y, z)$. We enumerate all solutions of the following Γ-formula:

$$\varphi' =_{def} \varphi \wedge \bigwedge_{I(x)=c} \psi_S(x_{ab}, x_b, x) \wedge \bigwedge_{I(x)=\{a,b\}} \psi_S(x, x_b, x_c),$$

where x_b and x_c are new variables and $x_{ab} \in \mathrm{Var}(\varphi)$, such that $I(x_{ab}) = \{a, b\}$. Since $J(x_b) = b$ and $J(x_c) = c$ for all solutions J of φ', there is a one-to-one correspondence between solutions of φ compatible with I and solutions of φ'. For every enumerated solution J of φ' we print $J|_{\mathrm{Var}(\varphi)}$. If $\{a, b\} \times \mathrm{C}_c \in \langle\Gamma\rangle_{\not\equiv,\neq}$ we can construct a similar Γ-formula and proceed in the same way. Since Γ is efficiently enumerable this algorithm works with polynomial delay. Hence, Γ is efficiently $E \to D$-enumerable. □

We presented new enumeration algorithms and gave criteria to detect constraint languages for which they can applied efficiently. Our example shows that these algorithms cover more cases than previously known from [Coh04]. In [SS06a] we give another algorithm that enumerates the efficiently only for very special constraint languages. The concept of partial enumerability allows to combine this and other efficient algorithms to new ones.

We achieved a dichotomy for constraint languages over the three-element domain that are conservative in a certain sense. The next step toward a full classification for the three-element domain, is to develop methods for achieving negative complexity results. Another topic for future research is to formulate the criteria given in Section 6.3 in terms of partial polymorphisms and strong partial clones.

Bibliography

[BHRV02] E. Böhler, E. Hemaspaandra, S. Reith, and H. Vollmer. Equivalence and isomorphism for Boolean constraint satisfaction. In *Computer Science Logic*, volume 2471 of *Lecture Notes in Computer Science*, pages 412–426, Berlin Heidelberg, 2002. Springer Verlag.

[BHRV04] E. Böhler, E. Hemaspaandra, S. Reith, and H. Vollmer. The complexity of Boolean constraint isomorphism. In *21st Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science, Berlin Heidelberg, 2004. Springer Verlag.

[BK05] C. Bazgan and M. Karpinski. On the complexity of global constraint satisfaction. In *16th International Symposium on Algorithms and Computation*, Lecture Notes in Computer Science, pages 624–633. Springer Verlag, 2005.

[BKJ00] A. A. Bulatov, A. A. Krokhin, and P. G. Jeavons. Constraint satisfaction problems and finite algebras. In *27th International Colloquium on Automata, Languages and Programming*, pages 272–282, 2000.

[BKKR69] V. G. Bodnarchuk, L. A. Kalužnin, V. N. Kotov, and B. A. Romov. Galois theory for Post algebras. i, ii. *Cybernetics*, 5:243–252, 531–539, 1969.

[BRSV05] E. Böhler, S. Reith, H. Schnoor, and H. Vollmer. Bases for Boolean co-clones. *Information Processing Letters*, 96:59–66, 2005.

[Bul06] A. A. Bulatov. A dichotomy theorem for constraint satisfaction problems on a 3-element set. *Journal of the ACM*, 53(1):66–120, 2006.

[CH96] N. Creignou and M. Hermann. Complexity of generalized satisfiability counting problems. *Information and Computation*, 125:1–12, 1996.

[CH97] N. Creignou and J.-J. Hébrard. On generating all solutions of generalized satisfiability problems. *Informatique Théorique et Applications/Theoretical Informatics and Applications*, 31(6):499–511, 1997.

[CHS07] P. Chapdelaine, M. Hermann, and I. Schnoor. Complexity of default logic on generalized conjunctive queries. In *9th International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 58–70, 2007.

[CKZ05] N. Creignou, P. G. Kolaitis, and B. Zanuttini. Preferred representations of boolean relations. *Electronic Colloquium on Computational Complexity (ECCC)*, (119), 2005.

[Coh04] D. Cohen. Tractable decision for a constraint language implies tractable search. *Constraints*, 9(3):219–229, 2004.

[Coo71] S. A. Cook. The complexity of theorem proving procedures. In *Proceedings 3rd Symposium on Theory of Computing*, pages 151–158. ACM Press, 1971.

[CZ06] N. Creignou and B. Zanuttini. A complete classification of the complexity of propositional abduction. *SIAM Journal on Computing*, 36(1):207–229, 2006.

[FV98] T. Feder and M. Y. Vardi. The computational structure of monotone monadis SNP and constraint satisfaction: a study through Datalog and group theory. *SIAM Journal on Computing*, 28(1):57–104, 1998.

[Gei68] D. Geiger. Closed systems of functions and predicates. *Pacific Journal of Mathematics*, 27(2):228–250, 1968.

[GJ79] M. R. Garey and D.S. Johnson. *Computers and intrctability. A guide to the theory of NP-completeness.* W. H. Freeman and Company, 1979.

[Got92] G. Gottlob. Complexity results for nonmonotonic logics. *Journal of Logic and Computation*, 2(3):397–425, 1992.

[Hem04] E. Hemaspaandra. Dichotomy theorems for alternation-bounded quantified boolean formulas. *CoRR*, cs.CC/0406006, 2004.

[JCG97] P. G. Jeavons, D. A. Cohen, and M. Gyssens. Closure properties of constraints. *Journal of the ACM*, 44(4):527–548, 1997.

[Jea98] P. G. Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200:185–204, 1998.

[JPY88] D. Johnson, C. Papadimitriou, and M. Yannakakis. On generating all maximal independent sets. *Information Processing Letters*, 27(3):119–123, 1988.

[Kar72] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, 1972.

[KS91] H. A. Kautz and B. Selman. Hard problems for simple default logics. *Artificial Intelligence*, 49(1-3):243–279, 1991.

[Lad75] R. Ladner. On the structure of polynomial-time reducibility. *Journal of the ACM*, 22:155–171, 1975.

[MS72] A. R. Meyer and L. J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential time. In *13th Symposium on Switching and Automata Theory*, pages 125–129. IEEE Computer Society Press, 1972.

[NJ04] G. Nordh and P. Jonsson. An algebraic approach to the complexity of propositional circumscription. In *19th Symposium on Logic in Computer Science (LICS)*, pages 367–376, 2004.

[Nor05] G. Nordh. A trichotomy in the complexity of propositional circumscription. In *11th International Conference on Logic for Programming*, volume 3452 of *Lecture Notes in Computer Science*, pages 257–269. Springer Verlag, 2005.

[NZ05] G. Nordh and B. Zanuttini. Propositional abduction is almost always hard. In *19th International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 534–539, 2005.

[Pap94] C. M. Papadimitriou. *Computational complexity*. Addison-Wesley, Reading, Massachusetts, 1994.

[Pos41] E. Post. The two-valued iterative systems of mathematical logic. *Annals of Mathematical Studies*, 5:1–122, 1941.

[Rei80] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1-2):81–132, 1980.

[Rom81] B. A. Romov. The algebras of partial functions and their invariants. *Cybernetics and Systems Analysis*, 17(2):157–167, 1981.

[Sch78] T. J. Schaefer. The complexity of satisfiability problems. In *10th Symposium on Theory of Computing*, pages 216–226. ACM Press, 1978.

[SS06a] H. Schnoor and I. Schnoor. Enumerating all solutions for constraint satisfaction problems. Technical report, Theoretical Computer Science, University of Hannover, 2006.

[SS06b] H. Schnoor and I. Schnoor. New algebraic tools for constraint satisfaction. In N. Creignou, P. Kolaitis, and H. Vollmer, editors, *Complexity of Constraints*, Dagstuhl Seminar Proceedings, 2006.

[SS07a] H. Schnoor and I. Schnoor. Enumerating all solutions for constraint satisfaction problems. In *24th International Symposium on Theoretical Aspects of Computer Science*, pages 694–705, 2007.

[SS07b] H. Schnoor and I. Schnoor. Partial polymorphisms and constraint satisfaction problems. In N. Creignou, P. Kolaitis, and H. Vollmer, editors, *Complexity of Constraints*, LNCS series, 2007. To appear.

[Sti90] J. Stillman. It's not my default: The complexity of membership problems in restricted propositional default logics. In *8th National Conference on Artificial Intelligence (AAAI)*, pages 571–578, 1990.

[Sto77] L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[Tur36] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265, 1936.

[Val79a] L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

[Val79b] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal of Computing*, 8(3):411–421, 1979.

Wissenschaftlicher Werdegang

Name	Ilka Schnoor geb. Johannsen
Geburt	11.09.1979 in Eckernförde
August 1986 - Juni 1999	Besuch der Freien Waldorfschule Eckernförde
Juli 1999	Abitur
Oktober 1999 - Februar 2005	Studium des Faches Mathematik mit Studienrichtung Informatik an der Leibniz Universität Hannover
März 2002	Vordiplom
Februar 2005	Diplom
Seit April 2005	Promotionsstudium des Faches Informatik an der Leibniz Universität Hannover
März 2005 - April 2007	Wissenschaftliche Mitarbeiterin am Institut für Theoretische Informatik, Leibniz Universität Hannover